# 概要

## 基本

データ収集

特徴量とラベル（正解）からなる入力と出力のペア。

モデルを選択する

線形回帰、ロジスティック回帰、決定木、ランダムフォレスト、ニューラルネットワークなど。

前処理

特徴量の標準化、カテゴリカル変数のエンコードなどが含まれます。

モデル学習

トレーニングデータを使用してパラメータを最適化する必要があります。

モデル評価する

学習済モデルをテストデータで評価。損失関数（例えば、平均絶対誤差、交差エントロピーなど）を使用。

モデル使用する

最後に、学習されたモデルを使用して新しいデータに対する推論を行います。

手法

重みとバイアスを適度かつ小さな乱数値を割り当て、2回以上結果を比較。



image classification

画像全体で物体を捉え、カテゴリ分けする。

object detection

画像中に特定の物体が存在するかどうか。

特に複数の物体が特定できる場合はmulti object detection と言われる事もある

## ニューラルネットワーク基本

畳み込み；convolution layer

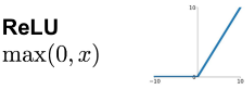
認識対象のデータの特徴を捉える手法。

データに応じて「特徴（パターン）」を見つけ、そのパターンとの一致度合いの「目印」を付けていく。一致すれば1、一致しなければ-1を付け、その特定範囲の平均値を最終的な目印値とする。

3×3の畳み込みをconv3などと表現する。

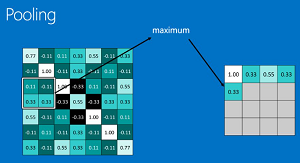
活性化；normalization　線形関数？

演算的な負担を取り除く為の処理？

ReLUsが良く使われる。

0以下なら「0」、0より上なら「入力値と同じ値」を返す。

学習効率が良く、勾配消失が発生しない。



プーリング；pooling layer

畳み込みで付けた「目印」を特定の範囲毎に確認し、範囲内の複数の値を間引く事で、縦横画素数を減らす手法。「特徴」だけはそのままに、画像サイズを小さくするイメージ。

max\_pool 最大値プーリング。基本的にはこれを使う。

avg\_pool 平均値プーリング。GANなどで使用。

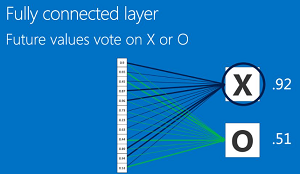
補足：

大きすぎるフィルターによる畳み込みはあまり意味がない。例えば9×9フィルターの畳み込み結果とほぼ同等のものを3×3フィルターの畳み込みで得られるうえ、こちらの方が計算量は大幅に減る。

この3つを基本とした様々な組み合わせで、画素数は減らしていきパターンを増やしていく。

この時の各処理の状態は「横画素数×縦画素数×パターン数」で表現される。

⇒ この例では最終的に「横2×縦2×パターン3」の配列的な12個のパラメータで１つの画像を表現できるようになった。



全結合；fully connected layer

特徴データだけになった配列的データから、どこに特徴が表れるのかを判定する。右図では×の方が高スコアなので、×の画像だと判定されている。

全結合層は複数積み重ねると性能がよくなると言われている。

（動画；英語）<https://www.youtube.com/watch?v=FmpDIaiMIeA>

（ppt資料）<https://slideplayer.com/slide/14119395/>

## 用語（基本）

深層学習

Deep Learning

DNN；Deep Nural Networkともされる。

## メモ

1,2,3,4,5・2,3,3,3,4・1,1,1,1,11・3,3,3,3,3・

単位を割る　というのがけっこう重要

8mは4mの何倍？ 8m /4m = 2　m（メートル）をメートルで割る

交絡変数 都道府県データにおける人口など

偏相関係数

デンドログラム

クラスタリング

Association Rule

相関の発見と利用

・データは「ばらつき具合」と「ばらつきに対してどうか」が重要

→ ばらつきの大きいデータの平均はなんの役にもたたない！

・仮説と検証が重要

・データが大量にあれば良い訳でなく、データの背景を検証する必要がある。

データ＋背景

現場実践力　が一番重要

1day

助成額70% 厚労省の助成金 第四次産業革命スキル習得講座

平日夜間も可能

# データ

## 概要

一般的に「Garbage In, Garbage Out」と言われ、適切なデータ

画像関連

最終的には28~64四方の正方形サイズする必要がある。

方法としては、画像ファイルを切り抜くか、リサイズするか。

validation（検証データ）

ハイパーパラメータが適切であるか確認する為のデータ

補足：クロスバリデーション　訓練データのいくつかを検証データに使う手法。

補足

ハイパーパラメータ モデル学習の設定項目のこと。

（学習時に更新されるのは重みのみ）

test（テストデータ）

学習が完全に終わった後、モデルの評価を行う。

訓練データを使ってはいけない。（過適合が起こる）

## ラベル付け

### 概要

annotation（アノテーション）とも呼ばれる。

自力でできない事もないが、かなりの手間なので、通常アノテーションツールを使うのが普通。

対応表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | TFRecord | YOLO | PascalVOC | CreateML | coco | その他 | 領域対応 |
| labelImg |  | 〇 | 〇 | 〇 |  |  | × |
| LabelMe |  | 〇 | 〇 |  |  |  | 〇 |
| Labelbox |  |  | 〇 |  | 〇 | csv, json |  |
| CVAT | 〇 |  | ？ |  | ？ |  | 〇 |
| Vott | 〇 | 〇 | 〇 |  |  |  |  |
|  |  |  |  |  |  |  |  |
| モデル | Tensorflow  datasets  (モデルではないが) | YOLO | ImageNet  EfficientDet |  |  |  |  |

矩形のみ labelImg

セグメンテーション対応 labelme、CVAT

### ソフトウェア

#### labelImg

##### 概要

Web情報だとどうも一番人気っぽい。

入手方法 pip(labelImg)

動作環境 python+Qt (Qt5/Qt4；Qt5推奨)

ライセンス MIT

Github <https://github.com/heartexlabs/labelImg>

対応形式：PascalVOC、YOLO、CreateML

##### チュートリアル

###### 共通

・「Open Dir」で画像ファイルのディレクトリを指定

・「Change Save Dir」でアノテーションファイルの保存先を指定



・矩形作成はCreate RectBox（右図参照）または、メニュー：Edit→Create RectBox

・出力ファイル形式はを押すと変わる。

PascalVOC、YOLO、CreateMLの順に変わる。

###### PascalVOCの場合

labelimg作業ディレクトリを以下のように構成しlabelimgを起動

OpenDirでdataset\imgasesを指定。Change Save Dirでdataset\annotationsを指定。

dataset\

├ images

│ ├ P0001.JPG

│ ├ P0002.JPG

├ annotations 最初は空っぽで良い

↓　作業終了後のファイル

dataset\

├ images

│ ├ P0001.JPG

│ ├ P0002.JPG

├ annotations

│ ├ P0001.xml

│ ├ P0002.xml

補足：本来であれば以下のようなディレクトリ構成にすべき

dataset\

├ train 学習データ

│ ├ images

│ ├ annotations

├ validation ニューラルネットワークのパラメータの良し悪し確認の検証データ

│ ├ 同様にannotations、imagesディレクトリを格納

├ test 学習後に汎化性能を確かめる為に使う

│ ├ 同様にannotations、imagesディレクトリを格納

#### LabelMe

セグメント系に対応。json形式で出力

入手方法 Github：https://github.com/wkentaro/labelme

動作環境 Webブラウザ（Webサービス）／python+QT

ライセンス

公式 <http://labelme.csail.mit.edu/Release3.0/>

#### VoTT

##### 概要

Visual Object Tagging Tool。Microsoftが開発するツール。（無料）

Githubではあまり人気が無いみたい

入手方法 Github：<https://github.com/microsoft/VoTT>

Web版 [https://vott.z22.web.core.windows.net/#/](https://vott.z22.web.core.windows.net/" \l "/)

（クラウドストレージしかつかえないっぽい）

動作環境 Windows、Linux、Mac

ライセンス MIT

対応形式

Tensorflow

Caffe

動作環境

Windows Githubから実行ファイル（.exe）をダウンロード

Ubuntu NodeJSとNPMのインストールした後git clone → npm

参考：<https://qiita.com/f-sk/items/7db148abb6d2d312367d>

Mac Githubから実行ファイル？（.dmg）をダウンロード

.vott がプロジェクトファイルっぽい

##### チュートリアル

新規作成

Display Name：プロジェクト名

Security Token：セキュリティの設定。任意。

Source Connection：画像を読み込む時のフォルダ

Target Connection：ラベルを書き出す時のフォルダ

Video Settings：ビデオにラベルを付ける時のフレーム数

Tags：ラベル

エキスポート

　ラベル編集画面でボタンを押す

#### Labelbox

動作環境 Webブラウザ（Webサービス）

ライセンス Apache-2.0

#### CVAT

Computer Vision Annotation Tool

Intelが開発。（OpenCVはもともとIntelが開発）

動作環境 Docker＋Webブラウザ（Webサービス）

ライセンス MIT

その他

#### annotorious

Webページでラベル付けができるJavaScriptのライブラリ。

画像や動画などのメディアに対してマウスドラッグやタップなどの操作でラベル付けができる。Web開発者は自分のアプリケーションやサイトに組み込むことができます

ライセンス：Apache License 2.0

### 資料（ラベル付け）

<https://qiita.com/shu-yusa/items/d19ea57e3cf9c4dbdce2>

#### opencvを用いた方法？

[https://intellectual-curiosity.tokyo/2019/07/02/%E3%82%AA%E3%83%AA%E3%82%B8%E3%83%8A%E3%83%AB%E3%81%AE%E7%94%BB%E5%83%8F%E3%81%8B%E3%82%89%E3%83%87%E3%83%BC%E3%82%BF%E3%82%BB%E3%83%83%E3%83%88%E3%82%92%E4%BD%9C%E6%88%90%E3%81%99%E3%82%8B%E6%96%B9/](https://intellectual-curiosity.tokyo/2019/07/02/オリジナルの画像からデータセットを作成する方/)

## オーバーサンプリング

### 概要

特定のデータ画像を増ししたい場合などに使う。

例えば不良率が非常に低い現場のAI対応などのときに、不良データを増やす。

### SMOTE

Synthetic Minority Over-sampling Technique

### DASYN

### Borderline-SMOTE

### Safe-level SMOTE

## tflite\_model\_maker

### csv形式

・UTF-8 でエンコード

・画像ファイルと同レベルのディレクトリにCSVを配置

・

書式：セット、画像ファイルのパス、

例

car,0.1,0.1,,,0.3,0.3,,

Google Cloud Storage バケットを使う必要がある？

## tensorflow→tflite

#ラベル

train\_labels = [0, 0, 1, 1]

train\_labels = np.array(train\_labels) #np.array形式にしないと読み込めない

#画像を読み込む

train\_images = []

for i in range(4):

print("/img/img"+str(i+1)+".jpeg")

img\_path = "/img/img"+str(i+1)+".jpeg"

img\_raw = tf.io.read\_file(img\_path)

img\_tensor = tf.image.decode\_image(img\_raw)

img\_final = tf.image.resize(img\_tensor, [28, 28])

img\_gray = tf.image.rgb\_to\_grayscale(img\_final)

img\_squeeze = tf.squeeze(img\_gray)

train\_images.append(img\_squeeze) # np.arrayに追加していく

train\_images = np.array(train\_images) #np.array形式にしないと読み込めない

train\_images = train\_images / 255.0

情報元：https://qiita.com/atsuto/items/f4003b31969071ee4d85

## 書式

### Pascal VOC XML

#### 概要

#### チュートリアル

ディレクトリ構成

my\_proj

├ img\

│ ├ P0001.JPG

│ ├ P0001.xml

[ P0001.xml ]

<annotation>

<folder>img</folder>

<filename>0001.JPG</filename>

<path>D:\my\_proj\img\P0001.JPG</path>

<source>

<database>Unknown</database>

</source>

<size>

<width>4000</width> <!-- 元画像サイズ：幅 -->

<height>3000</height> <!--元画像サイズ：高さ -->

<depth>3</depth> <!-- 元画像で使われている色の数　白黒:1、カラー:3 -->

</size>

<segmented>0</segmented>

<object> <!-- 対象物 -->

<name>crimped</name> <!-- ラベル名 -->

<pose>Unspecified</pose>

<truncated>0</truncated>

<difficult>0</difficult>

<bndbox> <!-- ラベルの位置を示すバウンディングボックス -->

<xmin>1443</xmin>

<ymin>584</ymin>

<xmax>1623</xmax>

<ymax>775</ymax>

</bndbox>

</object>

</annotation>

公式資料：但し読みにくい

<http://host.robots.ox.ac.uk/pascal/VOC/voc2012/guidelines.html>

## データセット

### 有名データセット

#### iris\_data

scikit-learnの場合

from sklearn import datasets

iris\_data = datasets.load\_iris()

# 説明変数

data = iris\_data.data # 後述する4つの項目のデータが入っている

# 目標変数

target = iris\_data.target # クラス番号0~2までが入っている

さらにpandasのDataFrameにしたい場合

import pandas as pd

# 以下のどちらでもよいらしい

df = pd.DataFrame(

data=iris\_data['data'],

columns= iris\_data ['feature\_names']

)

df = pd.DataFrame(

data=iris\_ iris\_data.data,

columns= iris\_data.feature\_names,

)

データ（説明変数）

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Sepal Length | がくの長さ |  |
| Sepal Width | がくの幅 |  |
| Petal Length | 花びらの幅 |  |
| Petal Width | 花びらの幅 |  |

行数：150

目的変数

|  |  |  |
| --- | --- | --- |
| 0 | Setosa |  |
| 1 | Versicolor |  |
| 2 | Virginica |  |

（公式）[https://scikit-learn.org/stable/modules/classes.html#module-sklearn.datasets](https://scikit-learn.org/stable/modules/classes.html" \l "module-sklearn.datasets)

Seabornを用いる場合

import seaborn as sns

df = sns.load\_dataset('iris')

#### MNIST

##### 概要

エムニスト

こんな方法もあるらしい

from tensorflow.examples.tutorials.mnist import input\_data

mnist = input\_data.read\_data\_sets("MNIST\_data/", one\_hot=True)

##### Fashion MNIST

TensorFlowで使う例

import tensorflow as tf

fashion\_mnist = tf.keras.datasets.fashion\_mnist

(train\_images, train\_labels), (test\_images, test\_labels) = fashion\_mnist.load\_data()

こんな方法もあるらしい

from tensorflow.examples.tutorials.mnist import input\_data

mnist = input\_data.read\_data\_sets("MNIST\_data/", one\_hot=True)

補足：tensorflow 2.8.2

#### CIFAR-10

別名：CIFAR10

近年の画像認識ネットワークのベンチマーク的なデータセット。

32×32 のカラー画像、その名の通り10クラス

### scikit-learn

from sklearn.datasets import fetch\_openml

|  |  |  |  |
| --- | --- | --- | --- |
| データセット名 | method名 | shapes | 備考 |
|  |  |  |  |
|  | load\_boston |  |  |
| ウィスコンシン乳がんデータセット | load\_brest\_cancer | (569, 30) |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

APIなどの詳細は別ドキュメント「python」へ

### mglearn

人工的で単純なデータセット

|  |  |  |
| --- | --- | --- |
| method名 | shapes | 備考 |
|  |  |  |
| make\_forge |  |  |
| make\_wave |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### e-Stat

日本の統計情報

https://www.e-stat.go.jp/

### Kaggle

[Dogs vs. Cats](https://www.kaggle.com/datasets/karakaggle/kaggle-cat-vs-dog-dataset)

<https://www.kaggle.com/datasets/karakaggle/kaggle-cat-vs-dog-dataset>

Ants & Bees

https://www.kaggle.com/datasets/gauravduttakiit/ants-bees

### MVtec AD

Hallconを作っている会社のデータセット

<https://www.mvtec.com/company/research/datasets/mvtec-ad>

### TensorFlow Hub

<https://tfhub.dev/>

### USI

http://archive.ics.uci.edu/ml/index.php

#### 車のデータセット

http://archive.ics.uci.edu/ml/datasets/automobile

### COCO 2017

https://cocodataset.org/index.htm#home

80クラス分類、カラー(3色)

### OpenImages

Googleが運営するデータセットのWebページ

https://storage.googleapis.com/openimages/web/index.html

python でダウンロード できなかった

https://cloud.google.com/blog/ja/topics/developers-practitioners/serving-data-cloud-storage

gs: でクラウド上のファイルを指定できる？

→ローカルのファイルは使えないのか？2022-12-10

[ bash ]

gsutil cp gs://cloud-ml-data/img/openimage/csv/salads\_ml\_use.csv ./salads\_ml\_use.csv

補足：gs: はGoogle Cloud Storageのバゲット（保管場所）の指定

tfds でダウンロードが可能？

dataset = tfds.load(‘open\_images/v7’, split='train')

for datum in dataset:

image, bboxes = datum["image"], example["bboxes"]

### Kaggle

登録が必要

### 画像データ

#### Image Net

<https://www.image-net.org/>

#### CIFAR-10

物体カラー写真（乗り物や動物など）の画像データセット

<https://www.cs.toronto.edu/~kriz/cifar.html>

### その他

#### The Oxford-IIIT Pet Dataset

<https://www.robots.ox.ac.uk/~vgg/data/pets/>

#### NewsAggregatorDataset

<https://archive.ics.uci.edu/ml/machine-learning-databases/00359/NewsAggregatorDataset.zip>

#### Md.Lab

提供元は大学が多い為、用途は非営利に限定される

<https://phy-lum.com/opendata/machine-learning-dataset.html>

### 資料

#### 公開データセット

20BN-JESTER

Atomic Visual Actions

BDD100K

CelebA

CIFAR-10/100

COIL-100

FaceForensics

Food-101

Google-Landmarks

Google/Open Images Dataset

HDR+ Burst Photography Dataset

ImageNet

iNaturalist

Kinetics

Large-scale Fashion Database

MNIST

Moments in Time Dataset

SALSA Dataset

STAIR Actions

TRECVID datasets

UCF101

The Oxford-IIIT Pet Dataset

YouTube-8M

YouTube-BoundingBoxes

#### リンク

データセット集１：<https://phy-lum.com/opendata/machine-learning-dataset.html>

# モデル学習

## モデル

### 概要

大きく学習のモデル（アルゴリズム）は以下の分類がある

分類

|  |  |  |
| --- | --- | --- |
| 名前 | 英語 |  |
| 教師あり | Supervised Learning | 入力値（説明変数）と正解（目的変数）のセットを与え、「識別器」を生成して、未知のデータにも適応させる。  クラス分類と回帰(regression)に大別される  タスクは「予測」 |
| 教師なし | Unsupervised Learning | 正解の分からない入力データの特徴を捉えて、データの傾向を分析させる。  パターンが分かってなかったり、常に変動したりする場合に向く  異常検知、グループ分けなどに向く。  タスクは「発見」 |
| 統計 |  | 外れ値、エンベロープ、マハラノビス距離 |
| 深層学習 |  |  |
| 強化学習 | Reinforcement Learning | 試行錯誤と「報酬」によって、各選択肢の価値を設定していき、最良の結果を得るための手法を導き出す。解の空間が広い場合に選択する。 |
|  |  |  |

答えを得るための「真の関数」が得られない為に、仮設の内汎化誤差が一番少ないものを仮の関数として使う。

ほとんどのアルゴリズムで以下の２つが重要になる：

特徴量エンジニアリング(feature engineering)

元の特徴量（＝データ）から新しい特徴量を作り出す事。

特徴量選択(feature selection)

学習で使用する特徴量（データ）からノイズを除き、有益なデータを抽出する事。

### データクラスタリング

#### k平均法

k-means clustering

クラスタ数kを指定して、すべてのクラスタ内のばらつきの和を可能な限り小さくしようとする。

観測点の割り当てにランダムな部分がある為、結果が少し異なる。その為、何回か実行して最もよく分離できたものを選択する。

#### 階層クラスタリング

Hierarchical Clustering

#### DBSCAN；Density-Based Spatial Clustering of Applications with Noise

### 機械学習系

#### KNN；k-nearest neighbors

k近傍法

neighborhood-based method（近傍ベース法、別名：遅延学習器；lazy learner）というアルゴリズムの中で最も使われる手法。

近傍ベース法は、与えられたデータセット内の観測地から近い値を見つけ出す。回帰タスク、類似性の計算などに使われる。外れ値に対して弱い。

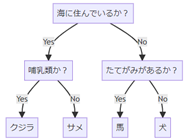
k近傍法はショッピングサイト（amazon）などにおける推薦システムで使われる。

パラメータ

近傍点の数 代替、3~5程度

データポイント間の距離

#### Decision Tree

別名：決定木

概要

データの空間をデータのラベルに基づいて多数の領域に階層化する。

予測のアルゴリズムがシンプルな為、比較的シンプルなデータに向く。

複雑なデータで扱う、または精度を重視する場合は他の分析方法が適切。

使用例

アンケート結果から将来の購買客層の行動予測を行う

注意点

分岐の数が非常に重要

訓練データに対して過剰適合しがちという欠点がある。

資料

参考：<https://helve-blog.com/posts/python/sklearn-classification-tree/>

決定木の種類：<https://qiita.com/3000manJPY/items/ef7495960f472ec14377>

#### RF；Random Forest

別名：ランダムフォレスト

データと特徴量をランダムにサンプリングする事で過剰適合を低減した決定木。

関連：bagging（別名：bootstrap aggregation）

用語

情報利得

データを分割する事でどれだけ「不純度」を下げられるか

不純度

ジニ不純度 各ノードに間違ったクラスが割り振られてしまう確率。

クラス数がNの場合

P1 本当はクラス0であるのにクラス1に分類されてしまう確率

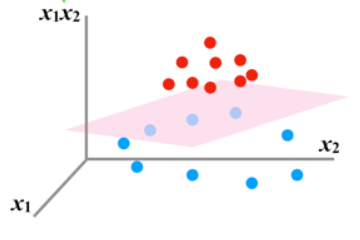
エントロピー 情報量を示す指標

分類誤差

利用方法：pydotplus

#### SVM；Support Vector Machine

サポートベクターマシン

各クラスの分割の為に新しい次元の境界線を引き、立体的に分離する。

境界付近にあるデータの誤分類を防ぐために分離超平面とデータとの最短距離が最も遠くなるように分離超平面を決定するアルゴリズム。境界は線形である必要は無い。

分類・回帰のみでなく、外れ値検出にも使える。

絶対値の大きな特徴量に極端に影響を受ける為、正規化による尺度揃えが必須。

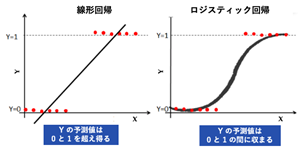
|  |  |
| --- | --- |
| 用語 | 意味 |
| サポートベクタ | 各クラスのデータの事 |
| 決定協会 | クラスを分離する境界線 |
| マージン | 別クラス間のデータの距離 |

利用方法：scikit-learn

#### One Class SVM

#### Logistic regression

ロジスティック回帰

別名：ロジスティック判別

クラス分類の為のアルゴリズム

いくつかの要因（説明変数）から「2値の結果（目的変数）」が起こる確率を説明・予測することができる統計手法。言い換えると、2つ以上のきれいに分ける事がる線形回帰がある状態の事。

例：大人と子供それぞれの身長、体重の関係

例：電子機器が故障るす場合、故障しない場合

変数間の相互作用を意図的に学習させたくないという問題設定の場合は、ロジスティック回帰を用いると良い

マーケティングでも使われる。ベルヌーイ分布に従う変数の統計的回帰モデルの一種。

#### 用語（機械学習系）

##### 混同行列Confusion matrix

|  |  |  |  |
| --- | --- | --- | --- |
| 総データ数  P+N | | 予測値 | |
| 陽性  (Positive;正例) | 陰性  (Negative;負例) |
| 実際 | 陽性 | TP  True Positive  正陽性 | FN  False Negative  偽陰性 |
| 陰性 | FP  False Positive  偽陽性 | TN  True Negative  正陰性 |

別名：error matrix

２値分類のタスクにおける評価指標の計算で使われる4つの値を並べた行列（表）。

評価指標

|  |  |  |  |
| --- | --- | --- | --- |
| 用語 | 別名 | 計算式 | 概要 |
| 感度  TPR | Sensitivity  TPR:True Positive Rate | TP / TP+FN | 負例を正しく判定できる確率 |
| 特異度  TNR | Specificity  TNR:True Negative Rate | TN / TN+FP | 正例を正しく判定できる確率 |
| PPV | Positive Predictive Value | 1.0 – FDR |  |
| NPV | Negative Predictive Value | 1.0 – FOR |  |
| FNR | False Negative Rate | 1.0 - TPR |  |
| 偽陽性率  FPR | False Positive Rate | 1.0 – TNR |  |
| FDR |  |  |  |
| FOR |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 用語 | 別名 | 計算式 | 概要 |
|  | Prevalence |  |  |
| 適合率 | Precision |  | 正例と予測したデータのうち、実際に正例であった割合。 |
| 再現率 | Recall |  | 実際の正例のうち、正例と予想したものの割合。 |
| 正解率  ACC | Accuracy |  |  |
| F値  F1 | F1 Score |  | 適合率と再現率の調和平均 |

補足

P 正例の特徴量数（≒データ個数）

N 負例の特徴量数（≒データ個数）

資料

wikipedia：<https://en.wikipedia.org/wiki/Confusion_matrix>

### 教師なし系

#### RBM；Restricted Boltzmann machine

制限付きボルツマンマシン

次元削減、分類、回帰、協調フィルタリング、特徴学習、トピックモデルなどに役立つ。

オートエンコーダ―には出力層があるが、RBNには無い

#### DBN；Deep Belief Network

RBNを多層に重ねたネットワーク。

特徴量検出器としてよく使われる。

### 原始的ニューラルネット

#### LeNet

1989年頃、畳み込みニューラルネットワークの最初の提案および実装。

畳み込みープーリングを2回，全結合層を3回

#### Alex Net

2012年のコンペで2位との大差をつけたことで深層学習を注目させるきっかけを作った有名なモデル。

論文：<https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>

### 画像分類(classifier)系

#### VGG

Visual Geometry Group

非常に有名なモデルで、フィルターサイズ3×3の畳み込みを繰り返す。派生？が６つあり、その内VGG16（派生C）とVGG19（派生E）が有名（らしい）。

CCPCCPCCCCPCCCCPCCCCP

（arxiv）<https://arxiv.org/abs/1409.1556>

Keras.applications.vgg16.VGG16

#### MobileNet

演算負荷を抑えて、従来並みの性能を持たせる事を目的としたモデル

#### DenseNet

#### EfficientNet

arxiv：<https://arxiv.org/abs/1905.11946>

モデルの「深さ」と「広さ」と「解像度(=入力画像の大きさ)」の3つをバランスよく調整する

精度もかなり高いらしい。

参考：<https://qiita.com/omiita/items/83643f78baabfa210ab1>

#### Inception

arxiv：https://arxiv.org/abs/1409.4842

別名：InceptionNet, GoogLeNet

### 物体検出系(detector)系

#### R-CNN

Region-based Convolutional Neural Network

画像セグメンテーションおよび物体検出のタスクを行うディープラーニングアルゴリズム

#### Fast R-CNN

#### Mask R-CNN

画像セグメンテーションおよび物体検出のタスクを行うディープラーニングアルゴリズム

Faster R-CNNを拡張したもの

#### SSD; Single Shot MultiBox Detector

１度のCNN演算で物体の「領域候補検出」と「クラス分類」の両方を行う。

画像分類のベースネットワークの全結合層を切り取り、シングルショットの畳み込み層を追加。

arxiv：<https://arxiv.org/abs/1512.02325>

#### EfficientDet

規模の拡張が容易で効率的な物体検出ネットワーク

概要

・EfficientNet（画像分類）を利用した物体検出（DETection）

arxiv：<https://arxiv.org/abs/1911.09070>

#### RetinaNet

One-Stage Detector は、リアルタイム処理に向いているとは言え、Two-Stage Detector と同等の検出精度が得られにくい、という問題に対して

ネットワーク構造の代わりに、目的関数の改良というアプローチを採用

#### DETR

Detection Transformer

Transformerを採用した最初のモデル。

オブジェクトの中心座標の推測が高精度。

### Residual Networkとその派生

#### Residual Network

##### 概要

ResNet。2015年のILSVRC & COCO コンペで優勝したモデル。1000層以上のニューラルネットワークの構成が可能。画像認識のモデルのベースラインとされており、2022年現在、近年のモデルはほぼこれの派生と言われるほど代表的なモデル。

なお、画像分類系（classifier）に分類される。

ソースコードなし

（arxiv）<https://arxiv.org/abs/1512.03385>

##### 使い方

PyTorchから <https://pytorch.org/hub/pytorch_vision_resnet/>

#### Inception ResNet v2

##### 概要

（arxiv）<https://arxiv.org/abs/1602.07261>

##### 使い方

PyTorchから

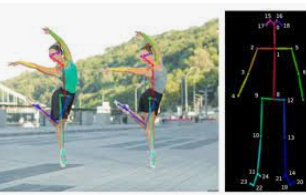
keras.applications.inception\_resnet\_v2.InceptionResNetV2(include\_top=True, weights='imagenet', input\_tensor=None, input\_shape=None, pooling=None, classes=1000)

### 姿勢推定

#### OpenPose

##### 概要

2D画像から骨格を検出する。

Git：<https://github.com/CMU-Perceptual-Computing-Lab/openpose>

開発言語：C++

利用可能な環境

##### チュートリアル（python)

学習済モデルをダウンロード

公式GitをCloneする。

（GithubのWebページでzipをダウンロードしても良い

# 公式Gitをclone

git clone https://github.com/CMU-Perceptual-Computing-Lab/openpose

# modelsディレクトリに移動

cd openpose/models

# ダウンロード　※計20分くらいかかる

# openpose/models/coco にダウンロードされる

sh getModels.sh # Linuxの場合

getModels.bat # Windowsの場合

ディレクトリ構成

├ openpose / 上でダウンロードしたGithubのclone

├ images / テスト画像を配置する

│ ├ sample.jpg テスト画像

├ venv / 仮想環境 opencv-pythonが必要

├ main.py

コンパイル＋インストールが必要らしい

##### チュートリアル（実行ファイル）

OpenPoseDemo.exe

--write\_json で実行結果をjsonに出力する事が可能

#### CPN

#### HRNet

OpenPoseのように骨格の検出ができる

### 画像領域(segmentation)系

#### 概要

対象の画像オブジェクトを線で区切るようなイメージ

セグメンテーションとも呼ばれる。

#### Auto Encoder

##### 概要

CNNを基本とした次元圧縮の手法。

エンコードして同じものにデコードする、という特徴を活かして画像特徴抽出。異常検知などで利用可能。

抽象的な特徴を自己学習。未学習の中間モデルに対応可能。

例えば、航空写真を入力として道路などの「領域」図を出力する事もできる。

（論文）<https://link.springer.com/chapter/10.1007/978-3-319-70096-0_39>

（arxiv）<https://arxiv.org/abs/2003.05991>

##### 使い方

TensorflowやPytorchで利用可能？

#### U－Net

Auto Encoderにさらに処理を加え、画像領域分割を行う。

画像の切り抜きなどにも使える

U-Net++、U2Netという派生モデルもある。

#### SegNet

複数の領域分解が可能。（Auto EncoderやU-Netは基本的に領域数2つ）

（arxiv）<https://arxiv.org/abs/1511.00561>

#### pix2pix

例えば白黒の写真を入力してカラー写真を出力するといった、

「画像間の特徴変換」を行う。

資料：<https://arxiv.org/abs/1611.07004>

#### Mask R-CNN

#### Panoptic FPN

### 時系列（再帰）系

#### RNN;Recurrent Neural Network

時系列、音声、言語、動画、グラフと言った**連続データを扱**うモデル。

再帰ニューロンの層を含み、「過去」の情報をもとに「未来」の値を予測するもの。

系列データ生成器。CNN層の情報を時間方向にも伝播する？

短時間内のデータの結びつきを見るのに向く？

#### LSTM；Long Short Term Memory

長期記憶にも対応したRNN。

#### Deep Envelope

時系列データに対して、**通常の傾向を学習**する。通常の傾向と外れたものを異常とする。

**複雑なパターン**下における正常／異常の判別が得意

補足：nonlinearitiesの共起表現；2つのデータ、2つの事象が同時に発生する傾向の事

#### Deep Autoencoder

ディープオートエンコーダ

**入力の再構築**を行うモデル。時系列データやグラフを入力として、正常時データを再構築できない場合に異常とみなす。**高次元のデータ**を扱うのに向く。

#### SNN；Spiking Neural Network

入力信号を時系列のバイナリ信号に変換して処理を行う。

SNNは加算演算のみで入出力関係を記述でき、脳の挙動に近く、かつ低消費電力。

NumPy と MatplotlibのみでSNNを実装

<https://compneuro-julia.github.io/_static/pdf/SNN_from_scratch_with_python_ver2_1.pdf>

参考資料：<https://arxiv.org/abs/2201.10879>

大学生？が作成

<https://snn.hirlab.net/>

torchでやるチュートリアル

<https://snntorch.readthedocs.io/en/latest/tutorials/tutorial_1.html>

#### 資料（時系列系）

https://www.tensorflow.org/tutorials/structured\_data/time\_series?hl=ja

### XAI関連

#### GAN

Generative Adversarial Network

敵対的生成ネットワーク

「ノイズ」を原料にGeneratorが「特徴」通りのデータを作成し、Discriminatorがその合否判定をする事でその識別／生成精度を高めていく手法。

#### StyleGAN

StyleGAN

PGGANに、AdaINという操作を取り入れることで画像の性質の制御が可能に

生成画像の品質も向上

StyleGAN2

AdaINとProgressive Growingを廃止。

StyleGAN2-ADA

データ拡張を確率的に行うadaptive discriminator augmentationを提案。

少数データでも学習が可能に。アーキテクチャはほぼStyleGAN2

StyleGAN3

エイリアシング対策を徹底することでテクスチャが固定される問題を解消

位置合わせされていない画像の生成も得意に

CycleGAN

PGGAN

#### DALL

#### Imagen

#### Parti

#### Midjourney

#### Stable Diffusion

#### Grad-CAM

別名：Gradient-weighted Class Activation Mapping

##### 概要

AIが識別の際に重要と思った箇所を色で表示してくれる

（arxiv）<https://arxiv.org/abs/1610.02391>

（Git）<https://github.com/ramprs/grad-cam>

##### 使い方

Kerasから

（公式）<https://keras.io/examples/vision/grad_cam/>

#### LIME

（資料）<https://arxiv.org/abs/1602.04938>

（Git）<https://github.com/marcotcr/lime-experiments>

#### Meaningful Perturbation

（資料）<https://arxiv.org/abs/1704.03296>

#### DeConvolution

転置畳み込み。逆畳み込み。

keras.Conv2DTranspose

### 統計学系

#### 因子分析

### 自然言語系

#### Transformer

自然言語処理におけるCNN、RNNの欠点を補うために誕生したアーキテクチャ。

ChatGPTやStable Diffusionでも用いられる。

IoTにおけるセンサデータのリアルタイム分析にも向く。

#### Self-Attension

文脈の中の特定の単語と他の単語との関係を捉える

### その他

#### VAE

Variational Auto Encoder

正常データだけを学習した異常検知

### 資料

#### Kerasで利用可能なモデル

Xception

VGG16

VGG19

ResNet50

InceptionV3

InceptionResNetV2

MobileNet

DenseNet

NASNet

MobileNetV2

情報元：<https://keras.io/ja/applications/>

#### 用語

##### One-Stage Detector

関心領域切り出しと物体の検出機能（カテゴライズ）を一度に同時に行う。

リアルタイム処理に向いている。

一般的に検出精度は低めと言われているが、近年精度が向上している。

モデル例：YOLO, SSD

##### Two-Stage Detector

関心領域切り出しと物体の検出機能（カテゴライズ）二段階に分けて行う。

比較的精度が高い。

モデル例：

##### 汎化誤差

generalization error

それまで見たことのないデータに対する機械学習アルゴリズム（モデル）の性能指標。

通常、これが最低となるようにハイパーパラメータ等を調整する。

## 目的別

### グラフや時系列データ

・deep autoencoder

・deep autoenvelope

・RNN

### 異常検出

### グループ分け

予想もつかなかった関連性が見えてくる場合がある。

### 推薦システム

協調フィルタリング こちらの方が広く用いられている

内容ベースフィルタリング

## 次元圧縮

### Autoencoder

オートエンコーダー；自己符号化器

3層ニューラルネットにおいて、入力層と出力層に同じデータを用いて教師なし学習させたもの。

**次元圧縮（削減）**のアルゴリズムの１つ。

新しく学習した表現を用いて、元の観測点を可能な限り近似する。

データの背後にある構造をとらえて元のデータの最も重要な性質だけを学習する為のデータを作成する。

PyTorchで利用可能

## 重み付け

### Influence-Balanced Loss

### Tomek links

アンダーサンプリングにより、識別曲線に近いデータのみ（もしくは片方のみ）を抽出して識別モデルを作成する。

from imblearn.under\_sampling import TomekLinks

tl = TomekLinks(return\_indices=True, ratio='majority')

X\_tl, y\_tl, id\_tl = tl.fit\_sample(X, y)

print('Removed indexes:', id\_tl)

plot\_2d\_space(X\_tl, y\_tl, 'Tomek links under-sampling')

情報元：<https://www.mlgraz.at/files/imbalanced.pdf>

## 活性化

### 概要

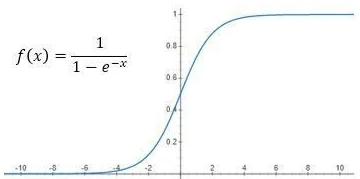
### 曲線

#### Sigmoid

別名：シグモイド関数、ロジスティック関数。

記号：ς（シグマ）。σも使われる。

名前の由来はς（末尾でのシグマ）に形が似ているから。



補足：aをゲインと呼ぶ。

a=1がよく用いられ、その場合「標準シグモイド関数」と呼ばれる。

神経細胞のモデル化。勾配消失が発生したり、層を重ねると学習量が小さくなり、学習が進まないこともある。

[ python ]

import numpy as np

def sigmoid(x):

return 1 / (1+ np.exp(-x)) # 補足：import numpy as np

#### Logit

シグモイド関数の逆関数

#### Softmax

別名：ソフトマックス関数

分類問題の出力層（最後の層）でつかわれる。

シグマの初期値がi=1になっているものもみられる

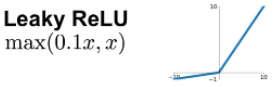
・関数の出力は0~1.0、**出力の総和が1**になる

⇒例えば、猫62%、犬35%、トラ3%といった感じになり便利。

・シグモイドとの違い

何次元でも0.0～1.0の間に収めてくれる関数。ニューラルネットワークでも用いられる。

scikit-learn：sklearn.utils.extmath.softmax



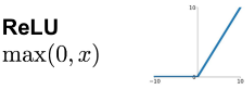




#### Linear

線形

#### ReLU；Rectified Linear Unit

0以下なら「0」、0より上なら「入力値と同じ値」を返す。

特徴

学習効率が良く、勾配消失が発生しない。**最も一般的**に用いられる活性化関数。

Pythonで書くと以下のようになる。

import numpy as np

def ReLU(x):

y = np.maximum(0, x)

return y

#### Tanh；Hyperbolic Tangent

-1.0～1.0の範囲に収めたSigmoid

#### ベジェ曲線

Bezier Curve

任意のポイントを通る曲線。Adobe Illustratorなどでドローツールとしてもつかわれる。

#### ガンマ関数

スターリングの近似

引数に対するn-1の階乗を返す。

#### ベータ関数

python

from scipy.special import beta

print(beta(7,28))

さまざまな曲線を描くことができる。

ベータ分布で使われる。

#### バスタブ曲線

例えば電子機器の故障率が時間経過とともにどう変化するかを調べる、といったたぐい。

#### 誤差関数

「正規分布のこの範囲内に収まる確率は何%か」を求める関数。

scipy：scipy.special.erf

### 恒等関数

こうとうかんすう；Identity function

別名：線形関数；Linear function

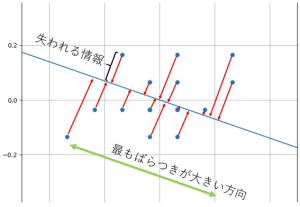
f(x)=x で表されるような簡単な関数。グラフ化すると直線になる。

身長と体重のようにある程度相関性が有る場合に使う。

### 次元圧縮

#### PCA ；Principal Component Analysis

主成分分析



最も一般的に用いられる線形次元削減手法の１つ。相関が高い特徴量をまとめて、より少数のパラメータ情報の新しい座標系に変換する。

イメージとしては右図の赤線を無視し、青線だけを意識した場合、線形の関数でデータのほとんどが説明できる事になる。

これにより、多次元のデータをより少ない次元に圧縮し、線形に相関のない特徴量で表現する事ができる。

具体的用途

・市場調査：「商品の質」「価格」「接客態度」の内、どれが顧客満足度に最も影響しているかを調べる。

・人事：「これまでの実績」「上司からの評価」「部下からの評価」「部下の実績」で誰を上長にすべきか。

ただし、削減されたデータに重要なデータが含まれていないとも限らないため注意が必要。

作成の仕方

全てのデータを標準化

データの分散共分散行列（相関行列；変数間の相関を表す行列）の作成

分散共分散行列（相関行列）を分解し、固有値と固有ベクトルを得る

固有値の大きい順に固有値固有ベクトルを主成分として採用

用語

寄与率 この主成分だけで元のデータの何割を説明することができているかを表した数字

固有値 各主成分が含んでいる情報の大きさを示す指標。一般的に「固有値が1以上」ある主成分が、元のデータとの関連が深いとされます

分かる事

各主成分がデータ全体をどれくらい説明できているか知ることができる。

補足

固有ベクトル を満たすでないベクトル とλの組み合わせが存在する時、Aの固有ベクトルという。

|  |  |  |
| --- | --- | --- |
| 記号 | 意味 | 例 |
| A | n×n行列でないベクトル |  |
| λ | スカラー | 2 |

また、λの事を固有値という

### 誤差算出関連

過剰適合

「誤差」の値に過剰に反応した結果、推論の数式がおかしなことになる事。

AIのモデル作成では、いかに過剰適合を避けるか　がポイント

#### 最小二乗法

別名：MSE;Leaset Squares Method、最小自乗法

直線回帰をするための誤差を算出する。

f(x)=ax+b　の時

b=

補足

Covxy：xとyの共分散 σx：xの分散 y（アッパーバー）：yの平均

#### 勾配降下法

損失関数の微分を順番に行い、損失関数の最小値を求める手法。

補足：勾配降下法では多次元の関数（＝３層以上のネットワーク）の最小値を見つける事はできない。

#### 誤差逆伝播法

Back Propagation

#### 交差エントロピー誤差関数

Cross-Entropy Loss

特徴

誤差が大きい程関数の変動が大きいため学習効率が良い。

AIでは良く使われる。

Tk：真値 yk：推定値

#### 資料

http://arduinopid.web.fc2.com/N78.html

### 損失関数

「正解値」とモデルによる出力された「予測値」とのズレ（Loss）の大きさを計算するための関数

### オプティマイザ―

### モデル

別名：学習器、判別器

中間層を増やすことは、数式の曲線を１回曲げる事。

### 軽量化手法

#### 概要

蒸留(Distillation)

枝刈り(Pruning)

量子化(Quantize) float32の代わりにfloat8を使ったりしてサイズや処理速度を向上させる。

#### 量子化

Quantization aware training

post-training quantization

quantization aware training 量子化誤差などを自動的に修正して最適化する手法。

### 学習率減衰

深層学習において、学習がある程度進んだ場所で学習率を下げると、急激に精度が向上することが知られており、汎化性能向上の為に使われる。

関連資料：

<https://akichan-f.medium.com/学習率減衰-バッチサイズ増大とearlystoppingの併用で汎化性能を上げる-tensorflow2-0-2d8f2e3709f4>

### 出力層

#### 概要

複数クラス分類を行う為の最終的な出力層。

#### Softmax

ソフトマックス

#### Crossentropy

クロスエントロピー

#### Binary Cross Entropy

Sigmoidと組み合わせてよく用いられる

Cost(y, y’) = -’

#### CategoricalCrossEntropy

Cost(y, y’) = -

#### Square Error

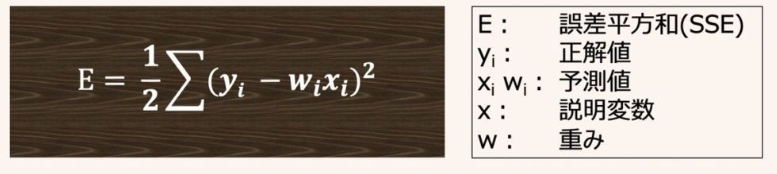
Cost(y, y’)

=

## 正則化

### 概要

過学習を抑える手法の内の１つ。「重み」としてのパラメータが大きくなりすぎた場合にペナルティを課す、という手法で



### L2正則化

損失関数f(x)との合計を最小にするパラメータwiを求める。

別名ユークリッド距離

### weight decay

過学習を防ぐための正則化手法の１つ。

学習中に重みの値を小さくする事で作成されるモデルの汎化性能向上を図る。

## 転移学習

別名：ファインチューニング、Transfer-Learning

学習済モデルの深い層だけを再編集する事で別のモデルを作成する手法

ディープラーニングでは浅い層ほどエッジなど汎用的な特徴が抽出され、深い層ほど訓練データに特化した特徴が抽出される傾向にある。これを利用して最下層（FC；Full Connected Layer）のみ学習をし直す事を転移学習と呼び、非常に効率が良いので頻繁に用いられる。

以下のモデルが転移元としてよく使われる（らしい）

VGG16：1000クラス分類

## モデル作成（学習）

### ハイパーパラメータ

#### 概要

|  |  |
| --- | --- |
| 項目 | 説明 |
| エポック数 | 訓練の繰り返し回数。一つの訓練データを何回繰り返して学習させるか。  大：過剰適合が発生しやすい  小：収束する前に終わってしまう |
| バッチサイズ | 一度にモデルに入力するデータの数。言い換えると重みを１度更新する際に用いる観測点の数。また演算で使用するメモリ数はこの値に比例する。  大：汎用的なモデルを作成できる  小：過剰適合が発生しやすい |
| cost function  コスト関数 | 予測値と実際の値との差異を評価し、モデルのパラメータを調整する為にもちいられる |
| optimizer  最小化アルゴリズム | コスト関数や重み、バイアスを調整する為のアルゴリズム。  勾配降下法、Adamなどがある |
| 重みの初期化方法 |  |
| weight decay | 重みの更新量を調整する。（正の値）  新しい損失 = 元の損失 + weight\_decay ×(L1/L2ノルム)  補足：すべてのパラメータに影響を与えるが、全てが等しく重要であるわけではないため、必ずしも最適な正則化手法とは限らない  大：重みの値は小さくなり、ゆっくりと収束する。学習不十分になりやすい。  小：過剰適合が発生しやすい |
| 学習率 | 重みを更新する際のステップのサイズ  大：収束しない。  小：収束が遅くなる |

Epoch

別名：エポック数

#### バッチサイズ

別名など：batch\_size

概要

勾配降下法※による誤差判定処理で、全データセット中の演算に使用する１グループのデータ数。

※損失関数の微分で誤差の最小値を調べ、重みの最適値を見つける手法

また、このようにグループ分けして重み更新を行う学習を「**ミニバッチ学習**」と呼ぶ。

目安としてtrainデータ数の1割ほど？

|  |  |  |
| --- | --- | --- |
|  | 長所 | 短所 |
| batch size  大 | ・学習が早く終わる | ・データの個性が失われる  ・高スペックのマシンが必要 |
| batch size  小 | ・１つ１つのデータの重要性が上がり、高精度なモデルが作成できる。  ・学習時のメモリが少なくて済む | ・ノイズの影響を受け、学習が収束しにくい |

備考

・深層学習のハイパーパラメータの中では最も基本的かつ重要なパラメータ。（らしい）

・１回のバッチ学習が終わると、すべての層の重みが更新される。

・慣例として2のn乗の値が使われるが、他の値でも良い。32, 64, 128, 256, 512, 1024, 2048あたりが一般的。

・一般的には小さい値から始めるのが良い。また、最大でも4096程度で留めておく方が良い。

・バッチサイズn=1 にした場合はオンライン学習と呼ばれる

・ベイズ最適化（ＢＯ）という、適切なバッチサイズを求めるパラメータ探索法がある。

#### エポック数

別名：学習回数、Max Epoch

概要

ミニバッチ学習を何回繰り返すか。理想値は「**損失関数（Loss Function）の値が収束**する回数」となる為、学習中の損失関数の推移をみるしか方法がない。

補足：学習がうまく行っている時は損失関数の値はだんだん小さくなる

|  |  |  |
| --- | --- | --- |
|  | 長所 | 短所 |
| epoch  大 | ・学習の際の精度が向上しやすい。 | ・データセットに過適合しやすい。  （overfitting） |
| epoch  小 | ・ | ・適合不足で学習が完了（収束）する。  (underfitting) |

備考

・短い時間でモデル性能を評価したい場合はエポック数を小さくする。いわゆるEarly Stopping。

・データの中にクラスがいくつ含まれるか、がこのパラメータを決める際で重要らしい。

・epoch数を増やせば増やすほど良い精度のモデルが作成できるわけでもない。

・学習が**完全に収束した場合は過学習**である可能性が高い。

#### イテレーション数

データセットに含まれるデータが少なくとも1回は学習に用いられるのに必要な学習回数。

バッチサイズが決まれば自動的に決まる。

例 全データ数4000の時にbatch size=400 → イテレーション=10

#### 重みの初期化方法

モデルの

ランダム初期化

ゼロ初期化

Xavier初期化

He初期化

#### 中間層の数

厳密にはハイパーパラメータではない。

#### 資料（ハイパーパラメータ）

バッチサイズ、イテレーション数、エポック数の説明；分かりやすい

<https://qiita.com/kenta1984/items/bad75a37d552510e4682>

### トラブルシューティング

学習モデルを作成する際に起こりやすい問題とその対応のメモ

#### 過学習

学習データに過度に適合してしまう事。

学習データには対応できるが、未知のデータには対応できなくなってしまう。

対策

・特にvalidationデータを増やす

・ハイパーパラメータの調整

バッチサイズ 増やす

weight decay 上げる

学習率 下げる

・正則化を行う

・early stoppingを検討する

#### モデルが不適切

問題の複雑さとモデル（アルゴリズム）が合っていない。

より複雑なモデルはもちろん、場合によっては非常にシンプルなモデルも検討する

#### early stoppingしてしまう

補足：モデル性能の向上が見られない為に学習演算を打ち切る事

・ハイパーパラメータを調整する

・パラメータ（patienceなど）を見直す

## 用語（モデル学習）

### ドロップアウト

dropout

過学習を抑制する手法。全結合層の各層の出力をある確率ratioratioでシャットアウトする

補足

アンサンブル学習 複数の弱学習器を統合して強識別機を作成する技術。

### 交差検証

モデルの構築と評価を複数回行う学習方式

### early stopping

学習中に長期間精度の改善が見られない時に中止する事。

train\_lossが改善していてもvalidation\_lossが改善しなくなった時点で学習を終了させる、という事が多い。

# 共通項目

## 自動最適化

### 概要

Auto Augmentation

画像データの回転、左右反転など手法で画像データ数を増やす手法。

（arxiv）<https://arxiv.org/abs/1805.09501>

### RandAugment

探索空間が小さく、高性能。

### Noisy Student

arxiv：<https://arxiv.org/abs/1911.04252>

## テンソル

### 概要

|  |  |  |  |
| --- | --- | --- | --- |
| テンソル | 変数で言うと | 概要 | 備考 |
| 0階テンソル | 普通の変数 | スカラー |  |
| 1階テンソル | １次配列 | ベクトル |  |
| 2階テンソル | 2次配列 | 行列 | matrix |
| 3階テンソル | 3次配列 | 行列を要素に持つ配列 | 行列が立体的にならぶイメージ |
| 4階テンソル |  |  | 3階テンソルに新たな軸を加える |
| 5階テンソル |  |  | 動画で用いる？ |

ベクトル同士の演算は、要素数が同じ時だけ成り立つ。それ以外の時はエラーとなる。

### ニューラルネットワーク

ニューラルネットワークの演算は、全て行列の演算。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 説明 | 入力層（第０層） | 第1層 | 第2層 |
| w | 重み |  |  |  |
| b | バイアス  （本来は縦ベクトル） |  |  |  |

このニューラルネットワークへの入力値（本来は縦ベクトル）とすると

第０層

これをシグモイド関数で活性化すると

第1層

これをシグモイド関数で活性化すると

第２層

さいごは恒等関数なのでそのまま

# パイプライン

## 概要

機械学習のモデル作成のライフサイクル管理ツール。

## MLflow

機械学習のライフサイクル管理（MLOps）を目的としたライブラリで、主に実験管理用途で使用される

## ML Pipelines

用語

MLOps

機械学習 Machine Learning (ML) と、ソフトウェア分野での継続的な開発手法である DevOps とを組み合わせた造語

## Beam

Apacheが運営するオープンソースのパイプライン？

## 資料

AIはなぜその答えを導き出したのか

<https://www.fujitsu.com/downloads/blog/jp/journal/2018-12-27-01.pdf>

### scikit-learnのチートシート

# 強化学習

## DeepMind

DeepMind Technologies社が作成した強化学習ソフト。2022年現在はGoogleにより買収。

後継プログラムであるAlphaGoはプロ囲碁棋士を破った事で有名

補足

各局面における「次の手」はチェスでは10120、将棋では10220、囲碁の探索量は10360乗存在する。

## 概要

多腕バンディット問題

方策勾配法

Sarsa

Q学習

DQN(Deep Q-network)

## 用語（強化学習）

|  |  |  |
| --- | --- | --- |
| 用語 | 説明 |  |
| エージェント | 環境に対して行動する主体 |  |
| 環境 environment | エージェントがいる世界 |  |
| 行動 action | エージェントのある状態における行動選択肢 |  |
| 状態 | エージェントの行動に対して更新される状態 |  |
| 報酬 reward | エージェントの行動に対する環境からの評価 |  |
| 方策 policy | エージェントが行動を決定する原理 |  |
| 即時報酬 | 行動直後に発生する報酬 |  |
| 遅延報酬 | 行動よりかなり遅れて発生する報酬 |  |
| 収益 | 即時報酬＋遅延報酬 |  |
| 価値 | エージェントの状態と方策を固定した場合の条件付きの収益 |  |

# ソフトウェア（AI系）

## 深層学習フレームワークの比較

|  |  |  |  |
| --- | --- | --- | --- |
| 指標 | Tensorflow | Keras | Torch |
| APIのレベル | 高と低 | 高 | 低 |
| 速度 | 速い | 遅い | 速い |
| 可読性 | 理解・使用しにくい | シンプル、簡潔で、理解しやすい | 複雑、理解しにくい |
| デバッグ | 困難 | シンプルなのでデバッグはほとんどない | 簡易 |
| データセット | 大規模のデータセット | 小規模データセット | 大規模のデータセット |
| 学習済みモデルの有無 | 有 | 有 | 有 |
| 学術での人気 | 1番 | 2番 | 3番 |
| 言語 | C++、CUDA、Python | Python | Lua、Python |

（参考）[https://www.tdi.co.jp/miso/tensorflow-keras-pytorch#37](https://www.tdi.co.jp/miso/tensorflow-keras-pytorch" \l "37)

特徴

Tensorflow

様々なオプション、実行環境など使える幅が広い

ドキュメントも豊富でトレーニングのサポートもある。

## 関連するpythonパッケージ

### scikit-image

勾配ブースティング

xgboost

lightgbm

クラスタリング

fastcluster

hdbscan pa

tslearn 時系列データのクラスタリング

## Tensorflow

### 概要

動作環境：Python3.6～3.9、Ubuntu 16.04以降、Windows7以降、macOS 10.12.6

Dockerのイメージも存在する

情報元：<https://www.tensorflow.org/install?hl=ja>

Bazelを用いたビルドも可能らしいが、pythonで使う事を前提としているらしい。

pythonでの利用

pip install tensorflow

パッケージ

tensorflow-gpu GPU搭載のPCで動かす際に必要となる（CUDAと共に使用）

### Raspberry Pi での使用

ラズパイのpipを使った方法では1.14.0までしかインストールできない。

公式サイトにインストール方法が記載されている：https://www.tensorflow.org/install/source\_rpi

OS：Kernel version: 5.15　Debian version: 11 (bullseye)　（2022年4/4版）

方法１：https://www.tensorflow.org/lite/guide/build\_cmake\_arm

8/8に試す

方法２<https://ichiken-usa.blogspot.com/2021/05/tensorflow230.html>

というか　https://itnext.io/installing-tensorflow-2-3-0-for-raspberry-pi3-4-debian-buster-11447cb31fc4

cython抜いて、python-dev-is-python2

方法３：<https://klab.hateblo.jp/entry/2022/03/19/153129>

公式ラズパイ検出

https://github.com/tensorflow/examples/tree/master/lite/examples/object\_detection/raspberry\_pi

### メモ

version2.0系で1.0っぽく使う方法

import tensorflow.compat.v1 as tf

tf.disable\_v2\_behavior()

学習中の表示

一例

Epoch 999/1000

1/1 [==============================] - 0s 88ms/step - loss: 0.2506 - accuracy: 0.8750 - val\_loss: 0.2500 - val\_accuracy: 0.8750

loss

accuracy

val\_loss

val\_accuracy

（参考）<https://sinyblog.com/deaplearning/keras_how_to/>

uint8のモデルの場合、全ての値を255で割る必要がある？ 2022-09-23

object\_detectorから各種クラスを用いる場合

from tflite\_model\_maker import object\_detector

spec = object\_detector.EfficientDetLite0Spec()

model\_specから文字列で指定する場合

from tflite\_model\_maker import model\_spec

spec = model\_spec.get('efficientdet\_lite0')

### チュートリアル

#### インストール

pip3 install tensorflow

pip3 install tensorflow-gpu # GPUがある場合

[ main.py ]

import tensorflow as tf

print(tf.reduce\_sum(tf.random.normal([1000, 1000])))

#### コンパイル

環境によってはpipなどでインストールするtensorflowがうまく動かない事もある。

その際はソースコードからコンパイルするのが遠回りなようで一番の近道。

##### Bezelのインストール

Bezelとは、CUDAランタイムがCUDAドライバを呼び出すためのAPIセットの事

<https://bazel.build/install?hl=ja>

debian系

bazel公式：<https://bazel.build/install/ubuntu?hl=ja>

sudo apt install bazel

##### 補足

公式ではcheckoutで2.2などのブランチへ移動しろ、とあるが、実際にはmasterしかない様子。

##### 資料

公式情報：https://www.tensorflow.org/install/source?hl=ja

#### 学習

ディレクトリ構成

├ dataset

│ ├ train

│ ├ test

#### モデル作成

##### 定義

データセットはFashionMNISTとし、

##### Tensorflow式

##### Keras式

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout

from tensorflow.keras.optimizers import SGD

model = Sequential()

model.add(Dense(256, # 隠れ層のニューロン数：256

input\_dim=784, # 入力層のデータサイズ：784

activation='relu')) # 活性化関数：ReLU

model.add(Dropout(0.5)) # ドロップアウト

model.add(Dense(10, # 出力層のニューロン数は10

activation='softmax')) # 活性化はソフトマックス関数

# モデルのコンパイル

learning\_rate = 0.1 # 学習率

model.compile( # オブジェクトのコンパイル

loss='sparse\_categorical\_crossentropy', # 誤差関数：スパース行列対応クロスエントロピー誤差

optimizer=SGD(lr=learning\_rate), # オプティマイザー：SGD

metrics=['accuracy'] # 学習評価として正解率を指定

)

model.summary() # ニューラルネットワークのサマリ（概要）を出力

#### モデルファイルのエキスポート

TFLITE\_FILENAME = 'efficientdet-lite-salad.tflite'

LABELS\_FILENAME = 'salad-labels.txt'

model.export(export\_dir='.', tflite\_filename=TFLITE\_FILENAME, label\_filename=LABELS\_FILENAME,

export\_format=[ExportFormat.TFLITE, ExportFormat.LABEL])

export\_format

公式：https://www.tensorflow.org/lite/api\_docs/python/tflite\_model\_maker/config/ExportFormat

### 基本概念

#### テンソル

≒配列変数と考える。均一型の多次元配列。numpyで言えばndarray。

immutableなので、中身の変更はできない。（つまりtuple）

#### shape

≒テンソル。Tensorflow内でテンソルを示すクラス？中身はtuple型（らしい）

なお、各クラスに存在するshapeメソッドは、各次元の要素数を返す。

#### 重み

自然な値（normal=正規分布）を定義する為の変数。具体的には、正規分布の分散の値などを設定する？

基本的に重み（≒分散）が大きければ大きいほど学習は失敗しやすい。

#### セッション(Session)

≒ターミナル。

構築されたデータフローグラフの演算処理を実際に行うランタイムへのクライアント。

### レファレンス（Tensorflow）

#### constant

import tensorflow as tf

rank\_0\_tensor = tf.constant(4)

float = tf.constant(0.08, dtype=tf.float32) # データ型に32bit 浮動小数点を指定

定数。「スカラー」または「階数 0」のテンソル。「軸」が存在しない。

比較：Variable

#### dtypes

##### dtypes.Dtype

tensorflowでサポートされている全テンソルを確認できる？

#### linalg

##### matmul

掛け算する

（公式）<https://www.tensorflow.org/api_docs/python/tf/linalg/matmul>

#### multiply

#### nn

##### 概要

ニューラルネットワークに関連したクラスをまとめたモジュール？

##### conv2d

二次元畳み込みを行う

構文：tf.nn.conv2d(**input,filters,strides,padding**,data\_format='NHWC',dilations=None,name=None)

例）2022.06.16あまり良い例じゃないので、差し替え予定

tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

（関連）depthwise\_conv2d, separable\_conv2d

引数

input

4次元以上のテンソル（配列）が必要

[batch, 縦, 横, チャンネル数 ]

補足：チャンネル

単色の場合： [ 値/255.0 ] のfloat配列

3色の場合：[R/255.0, G/255.0,B/255.0] のfloat配列

となる（と思う；2022年6月16日）

filter

重みを指定する？

(縦, 横, チャンネル数, フィルター数,)

補足：3×3、5×5、7×7など奇数な事が多い？

stride

畳み込みの時の歩幅。数画素ずつフィルタの適用範囲を計算するための値。

1,2,4次元のlist(int)。但し[1, stride, stride, 1]と先頭と最後は１固定。

padding

ピクセル数が奇数の時の挙動を指定する（畳み込みは2の倍数で行われる）。

以下の２つのどちらか：

SAME 埋める部分を全て0にする。

VALID

##### max\_pool

ksize カーネルサイズ。

##### relu

特定の値を最大値としたrelu活性化の演算を行う（？）

<https://www.tensorflow.org/api_docs/python/tf/nn/relu>

##### softmax

softmax活性化の演算を行う

構文：tf.nn.softmax( logits, axis=None, name=None)

このメソッドだけで、畳み込みから活性化まで出来る？？（セミナーより）

##### softmax\_cross\_entropy\_with\_logits

##### softmax\_cross\_entropy\_with\_logits\_v2

##### sparse\_softmax\_cross\_entropy\_with\_logits

#### reshape

構文：tf.reshape(**tensor, shape**, name=None)

tensor =-1で任意。

#### Variable

TesorFlow内の変数を扱うクラス

reprで表示される文字列？

<tf.Variable 'Variable:0' shape=(10,) dtype=float32\_ref>

'Variable:0' はカウントアップされていく。'Variable:1'、'Variable:2'…

（公式；英語）<https://www.tensorflow.org/api_docs/python/tf/Variable>

（公式ガイドより）<https://www.tensorflow.org/guide/variable?hl=ja>

#### zeros

全要素の値が0のテンソル（配列）を作成する

構文：tf.zeros(shape, dtype=tf.dtypes.float32, name=None)

公式：<https://www.tensorflow.org/api_docs/python/tf/zeros>

### レファレンス（Tensorflow 1.n）

#### placeholder

プレースホルダーを作成。データは未定のままグラフを構築し、具体的な値は実行する時に与える、といった事ができる。C言語のポインタに近い？

#### truncated\_normal

切断正規分布(truncated normal) から乱数を出力

補足：normal distributionは「正規分布」

（公式）<https://www.tensorflow.org/api_docs/python/tf/random/truncated_normal>

パラメータ

shape

行列（≒配列）の数を指定する

initial = tf.truncated\_normal(10000, stddev=0.1) # 平均0、分散0.1の正規分布された要素数10000の配列

variable = tf.Variable(initial) # 実際に配列（厳密にはtuple）を作成

補足：要素数が少ないと、きれいな正規分布になりにくい（らしい）

stddev

切断まえの正規分布の分散値。dtype？

stddev=0.1は「平均0、分散0.1の正規分布」を意味する。

#### assign

#### Session

##### 概要

shell（≒cmd）のような会話形式入出力の設定を行う？

補足

Sessionは元々「着席する」の意で、転じて「会議を始めてから終わるまで」になった。

##### InteractiveSession

tensorflow1系にしか存在しない？

tf.compat.v1.InteractiveSession(

target='', graph=None, config=None

)

### レファレンス（tensorflow lite）

https://www.tensorflow.org/api\_docs/python/tf/lite

#### 基本

##### detect

[ 2022-09-21現在　まだ不明点多い]

import numpy as np

from PIL import Image

from time import sleep

from tflite\_runtime.interpreter import Interpreter

# モデルファイル読み込み

interpreter = Interpreter(model\_path='models/some\_model.tflite')

# メモリ確保

interpreter.allocate\_tensors()

# 学習モデルの入力層・出力層のプロパティを取得( dictのlist [{}, ...] 形式)

input\_details = interpreter.get\_input\_details() # どのモデルを使用してもlistのlenは1

output\_details = interpreter.get\_output\_details() # 使用するモデルによってlistのlenは変わる

ftf

# 画像ファイルの読み込み（Pillowを使用）

height = input\_details[0]['shape'][1]

width = input\_details[0]['shape'][2]

img = Image.open('images/some\_image.png')

img = img.resize((width, height))

# 入力層のテンソルデータ構成の取得

input\_shape = input\_details[0]['shape']

# 読み込んだ画像から、テンソルデータを作成

input\_data = np.expand\_dims(img, axis=0)

# 入力テンソルの設定

interpreter.set\_tensor(input\_details[0]['index'], input\_data)

# 推論実行

interpreter.invoke()

# 推論結果の取り出し

output\_data = interpreter.get\_tensor(output\_details[0]['index']) # 検出結果を全て格納する

boxes = interpreter.get\_tensor(output\_details[0]['index'])[0] # 検出のバウンディングボックス

classes = interpreter.get\_tensor(output\_details[1]['index'])[0] # 分類されたラベル情報

scores = interpreter.get\_tensor(output\_details[2]['index'])[0] # 一致率

参考資料

<https://github.com/rianrajagede/object-detection/blob/master/scripts/TFLite_detection_image.py>

次ページに続く

動作確認できたモデル

・ Mobilenet SSD version2

以下のコマンドでダウンロード可能

ファイル名：mobilenet\_ssd\_v2\_coco\_quant\_postprocess.tflite

mkdir -p all\_models

wget https://dl.google.com/coral/canned\_models/all\_models.tar.gz

tar -C all\_models -xvzf all\_models.tar.gz

rm -f all\_models.tar.gz

・EfficientDet-Lite4

DL：<https://tfhub.dev/tensorflow/lite-model/efficientdet/lite4/detection/default/2>

補足

#### tflite\_model\_maker

##### 概要

pip install -q tflite-model-maker

##### DataLoaderクラス

from\_csv

from\_cache

from\_pascal\_voc

gen\_dataset

split

公式：<https://www.tensorflow.org/lite/api_docs/python/tflite_model_maker/object_detector/DataLoader>

##### export

学習済モデルから.tfliteファイルを作成する。

例

TFLITE\_FILENAME = 'efficientdet-lite-salad.tflite'

LABELS\_FILENAME = 'salad-labels.txt'

model.export(export\_dir='.', tflite\_filename=TFLITE\_FILENAME, label\_filename=LABELS\_FILENAME,

export\_format=[ExportFormat.TFLITE, ExportFormat.LABEL])

##### evaluate\_tflite

.tfliteファイルを読み込んで、評価を行う？

TFLITE\_FILENAME = 'efficientdet-lite-salad.tflite'

model.evaluate\_tflite(TFLITE\_FILENAME, test\_data)

##### object\_detector

概要

###### create

[ python ]

from tflite\_model\_maker import object\_detector

label\_map = {

1 : ‘fuga’,

2 : ‘hoge’,

}

train\_data = object\_detector.DataLoader.from\_pascal\_voc(train\_data\_dir, annotation\_data\_dir, label\_map=label\_map)

validation\_data = object\_detector.DataLoader.from\_pascal\_voc(val\_data\_dir, val\_annotation\_data\_dir, label\_map=label\_map)

model = object\_detector.create(

train\_data =

validation\_data =

epochs = 10,

batch\_size = 100,

train\_whole\_model = True,

)

|  |  |
| --- | --- |
| det\_loss: | オブジェクト検出損失；検出物体の存在するか否かを検出する損失関数。 |
| cls\_loss: | クラス予測に対する損失関数。 |
| box\_loss: | オブジェクトの位置情報を予測するための損失関数。 |
| reg\_l2\_loss: | モデルの重みを正則化するためのL2正則化損失。 |
| loss: | モデルの全体的な損失値。上記の損失関数の合計。 |
| learning\_rate: | モデルの学習率。大：学習が進むが収束しない　小：パラメータ更新はゆっくり |
| gradient\_norm: | バックプロパゲーション中に計算される勾配のノルムです。 |
| val\_det\_loss: | Validation時のオブジェクト検出損失。 |
| val\_cls\_loss: | validation時のクラス予測損失です。 |
| val\_box\_loss: | validation時の位置予測損失です。 |
| val\_reg\_l2\_loss: | validation時のL2正則化損失です。 |
| val\_loss: | validation時の全体的な損失値。上記の損失関数の合計。 |

補足：

L2正則化損失 重みが大きくなりすぎないように、モデルの重みに対して罰則を与える。

###### DataLoader

from\_cacheメソッド

こんな感じで使うらしいが？

train\_data = object\_detector.DataLoader.from\_cache('/tmp/cache/train')

from\_csvメソッド

from\_pascal\_vocメソッド

公式：[https://www.tensorflow.org/lite/api\_docs/python/tflite\_model\_maker/object\_detector/DataLoader](https://www.tensorflow.org/lite/api_docs/python/tflite_model_maker/object_detector/DataLoader" \l "from_csv)

#### model

###### evaluate

model = object\_detector.create(

～略～

)

test\_data = object\_detector.DataLoader.from\_pascal\_voc(test\_data\_dir, test\_annotation\_data\_dir, label\_map=label\_map)

# 評価実行

model.evaluate(test\_data)

|  |  |
| --- | --- |
| 名前 | 概要 |
| AP  (Average Precision)： | 各クラスについて、検出された物体の正解率の平均値。この値が高いほど、モデルの性能が良いと判断できます。 |
| AP50： | IOUが 0.5 以上のものだけを正解として計算する場合の平均精度(AP)。 |
| AP75： | IOUが 0.75 以上のものだけを正解として計算する場合の平均精度(AP)。 |
| APs： | 物体の面積が小さい物体（小さいサイズ）に対しての AP。 |
| APm： | 物体の面積が中くらいの物体（中くらいのサイズ）に対しての AP。 |
| APl： | 物体の面積が大きい物体（大きいサイズ）に対しての AP。 |
| ARmax1： | 1つの画像につき、最大で1つの物体が検出された場合の再現率。 |
| ARmax10： | 1つの画像につき、最大で10個の物体が検出された場合の再現率。 |
| ARmax100： | 一つの画像につき、最大で100個の物体が検出された場合の再現率。 |
| ARs： | 物体の面積が小さい物体に対しての再現率。 |
| ARm： | 物体の面積が中くらいの物体に対しての再現率。 |
| ARl： | 物体の面積が大きい物体に対しての再現率。 |
| AP\_/my\_class： | 特定のクラスの AP。たとえば、AP\_/my\_classはmy\_classの平均精度。 |

#### tflite\_runtime

##### Interpreter

from tflite\_runtime.interpreter import Interpreter

get\_tensor(index)

[https://www.tensorflow.org/api\_docs/python/tf/lite/Interpreter#get\_tensor](https://www.tensorflow.org/api_docs/python/tf/lite/Interpreter" \l "get_tensor)

set\_tensor

index何番にこのテンソル（配列）をセットする、という感じらしい。

[https://www.tensorflow.org/api\_docs/python/tf/lite/Interpreter#set\_tensor](https://www.tensorflow.org/api_docs/python/tf/lite/Interpreter" \l "set_tensor)

get\_objects

### レファレンス（Keras）

#### 概要

##### モデル作成２つの方法

layers（functional API）でモデルを作成するチュートリアル。

より柔軟に対応出来る。

<https://keras.io/ja/getting-started/functional-api-guide/>

Sequentialでモデルを作成するチュートリアル

簡単。

<https://keras.io/ja/getting-started/sequential-model-guide/>

#### applications

##### vgg16

#### dataset

##### 概要

文字通り、データセットを管理する

# Fashion-MNISTデータセットの読み込み

(x\_train, y\_train), (x\_test, y\_test) = tf.keras.datasets.fashion\_mnist.load\_data()

##### Flatten

入力をフラットにする（２次元データを一次元データに変換する）

tf.keras.layers.Flatten(input\_shape=(28, 28)), # 28×28サイズの2次元データを784の1次元データにする

（関連）Dense

（公式）[https://keras.io/ja/layers/core/#flatten](https://keras.io/ja/layers/core/" \l "flatten)

##### Dense

全結合ニューラルネットワークで、１つのニューラルネットワークを定義する。

tf.keras.layers.Dense(128, activation='relu'), # 出力層128、活性化関数reluのニューラルレイヤーを作成

引数

units 出力層の数。

activation 活性化関数を選択する。

input\_shape 入力層の次元数を指定。これを設定するのは最初の入力層だけ。

name ニューラル

（公式）[https://keras.io/ja/layers/core/#dense](https://keras.io/ja/layers/core/" \l "dense)

##### Softmax

##### Conv2Dメソッド

filters: フィルタ数

kernel\_size: フィルタのサイズ（補足：真ん中を作る為、奇数が使いやすい）

padding: "valid"か"same"のどちらか

data\_format

strides: 畳み込みの歩幅

<https://keras.io/ja/layers/convolutional/>

#### layers

##### 概要

1つ以上のテンソルを入力として受け取り、1つ以上のテンソルを出力する呼び出し可能なオブジェクトLayerを基本クラスとするクラス群（functional API）を構成するモジュール？

Sequentialクラスを使った方がはるかに楽。

##### Conv2D

2次元畳み込み層を提供

import tensorflow as tf

class CNN(tf.keras.Model):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# 畳み込み層1：活性化関数はReLU

# (バッチサイズ, 32, 32, 3) -> (バッチサイズ, 32, 32, 32)

self.conv2D\_1 = tf.keras.layers.Conv2D(

filters=32, # フィルターの数は32

kernel\_size=(3, 3), # 3×3のフィルターを使用

padding='same', # ゼロパディングを行う

input\_shape=(28, 28, 1), # 入力データの形状

activation='relu' # 活性化関数はReLU

)

# この場合 入力：(28, 28, 1) ⇒ 出力：(28, 28, 32)となる

（公式）https://keras.io/api/layers/convolution\_layers/convolution2d/

##### Dropout

class CNN(tf.keras.Model):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# ... 省略

self.dropput1 = tf.keras.layers.Dropout(0.5)

##### MaxPooling2D

class CNN(tf.keras.Model):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# ... 省略

# (バッチサイズ, 32, 32, 32) -> (バッチサイズ, 16, 16, 32)

self.pool1 = tf.keras.layers.MaxPooling2D(

pool\_size=(2,2) # 縮小対象の領域は2x2

)

##### Dense

出力層を提供する？

class CNN(tf.keras.Model):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# ... 省略

# 全結合層1：活性化関数はReLU

# (バッチサイズ, 4096) -> (バッチサイズ, 512)

self.fc1 = tf.keras.layers.Dense(512, activation='relu')

##### Flatten

入力テンソルが何であれ、1回テンソル（1次元配列）に変換する。

class CNN(tf.keras.Model):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# 例えばバッチサイズが (8, 8, 64) -> 8×8×64= (4096)

self.flatten = tf.keras.layers.Flatten()

#### losses

##### 概要

##### CategoricalCrossentropy

import tensorflow as tf

# マルチクラス分類のクロスエントロピー誤差を求めるオブジェクト

loss\_fn = tf.keras.losses.CategoricalCrossentropy()

#### metrics

4. 勾配降下アルゴリズムによるパラメーターの更新処理

import tensorflow as tf

# 損失を記録するオブジェクトを生成

train\_loss = tf.keras.metrics.Mean()

# カテゴリカルデータの精度を記録するオブジェクトを生成

train\_accuracy = tf.keras.metrics.CategoricalAccuracy()

#### model

#### optimizer

##### 概要

オプティマイザ；最適化アルゴリズムに関連したモジュール？

補足

clipnormとclipvalueはすべてのオプティマイザで有効な引数？

##### SGD

確率的勾配降下法

SGD ；Stochastic Gradient Descent,

連続最適化問題に対する勾配法の乱択アルゴリズム。

（公式）[https://keras.io/ja/optimizers/#sgd](https://keras.io/ja/optimizers/" \l "sgd)

##### Adam

概要

最適化アルゴリズムの大定番。

迷ったら、とりあえずAdam

最小化のアルゴリズム（？）にAdaGradとMomentumを組み合わせたアルゴリズムを用いる

参考資料：[https://developer.nvidia.com/blog/deep-learning-nutshell-history-training/#rmsprop](https://developer.nvidia.com/blog/deep-learning-nutshell-history-training/" \l "rmsprop)

（公式）[https://keras.io/ja/optimizers/#adam](https://keras.io/ja/optimizers/" \l "adam)

備考

・極端に大きな学習率になることから、SGDと比較して収束が安定しにくい

#### Sequential

##### 概要

モデルレイヤーの設定を行うクラス。

（公式）<https://keras.io/ja/getting-started/sequential-model-guide/>

方法は大きく２つ

空のmodelを作成して、addしていく方法

from keras import layers

from keras import models

model = models.Sequential()

model.add(layers.Dense(10, activation='relu', input\_shape=(4,)))

model.add(layers.Dense(3, activation='softmax'))

Sequentialクラスの\_\_init\_\_（コンストラクタ；イニシャライザ）でlist型で一気に設定する

import tensorflow as tf

model = tf.keras.Sequential([

tf.keras.layers.Flatten(input\_shape=(28, 28)),

tf.keras.layers.Dense(128, activation='relu'),

tf.keras.layers.Dense(10)

])

関連情報として、Sequentialの他にもう一つfunctional APIとともに用いるモデルクラス群が存在する。（らしい）

##### \_\_init\_\_

引数

input\_shape

##### predict

入力サンプルに対する予測値の出力を生成

構文：predict(x, batch\_size=None, verbose=0, steps=None)

引数

x: 入力データで，numpy 配列の形式．

batch\_size: 整数．指定しなければデフォルトで32になります．

verbose: 進行状況メッセージ出力モード，0または1．

steps:

評価ラウンド終了を宣言するまでの総ステップ数（サンプルのバッチ）。

デフォルト値のNoneならば無視。

##### compile

model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

metrics=['accuracy'])

引数

optimizer オプティマイザ；最適化アルゴリズム

モデルが表示するデータとその損失関数に基づいたモデルの更新方法

loss 損失関数。

metrics 指標。

（公式）[https://keras.io/ja/models/model/#compile](https://keras.io/ja/models/model/" \l "compile)

##### fit

構文：fit(x=None, y=None, batch\_size=None, epochs=1, verbose=1, callbacks=None, validation\_split=0.0, validation\_data=None, shuffle=True, class\_weight=None, sample\_weight=None, initial\_epoch=0, steps\_per\_epoch=None, validation\_steps=None)

引数

epochs エポック数

（公式）[https://keras.io/ja/models/model/#fit](https://keras.io/ja/models/model/" \l "fit)

##### evaluate

構文：evaluate(x=None, y=None, batch\_size=None, verbose=1, sample\_weight=None, steps=None)

引数

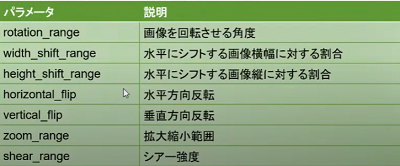
verbose 進行状況の表示モード。0 = 表示なし，1 = プログレスバー、2=

[https://keras.io/ja/models/model/#evaluate](https://keras.io/ja/models/model/" \l "evaluate)

#### preprocessing.image

##### ImageDataGenerator

Data Augmentation



### モデルの作成

#### tflite-model-maker

pip install -q tflite-model-maker # 補足：これだけでtensorflowなどもインストールしてくれる

### 識別

#### 概要

要素番号比較：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | mobilenet\_ssd | | | efficientDet | | |  |
|  | 要素番号 | name | index | 要素番号 | name | index |  |
| ボックス | 0 | なし | 252 | 1 | 3 | 598 |  |
| クラス | 1 | 1 | 253 | 3 | 2 | 599 |  |
| スコア | 2 | 2 | 254 | 0 | 1 | 600 |  |
| 物体数 | 3 | 3 | 255 | 2 | 0 | 601 |  |
|  |  |  |  |  |  |  |  |

⇒ index番号の若い順にボックス、クラス、スコア、物体数の順になるらしい。

#### interpreter.get\_input\_details()　の結果

公式：[https://www.tensorflow.org/lite/api\_docs/python/tf/lite/Interpreter#get\_input\_details](https://www.tensorflow.org/lite/api_docs/python/tf/lite/Interpreter" \l "get_input_details)

補足：get\_output\_details() で得られる結果も同じ

|  |  |  |
| --- | --- | --- |
|  | 説明 | 具体的な値の例 |
| 'name', | テンソルの名前 |  |
| 'index', | テンソルのindexget\_tensor | ０　　get\_tensor() の引数として使う値 |
| 'shape', | 各次元の要素数 | ([1, 1, 1, 3], dtype=int32) |
| 'shape\_signature', | shapeと同じだが、要素数不明の次元がある場合は-1で示される。 | ([ 1, -1, -1, 3], dtype=int32) |
| 'dtype', | numpyのデータ型 | numpy.uint8 |
| 'quantization', | per-tensor quantizationでのみ使われる？ | (0.0078125, 128), |
| 'quantization\_parameters', | テンソル量子化に使われるパラメータ？ | 割愛 |
| scales |  | ([0.0078125], dtype=float32), |
| zero\_points | ？ | ([128], dtype=int32) |
| quantized\_dimension | ？ | 0 |
| 'sparsity\_parameters' |  | {} |

#### pycoral/examples/detect\_image.py

Github：https://github.com/google-coral/pycoral

#### example-camera-master/opencv/detect.py

Github：https://github.com/google-coral/examples-camera

#### /usr/bin/edgetpu\_detect

Google coral（mendel Linux）インストール時に格納されているpythonスクリプト

### 関連パッケージ

#### tf\_slim

#### tflite-model-maker

tensorflow-lite用のモデルを簡単に作成できるようにしたパッケージ。

具体的にはMobileNetを元に転移学習を行う。（らしい）

（公式）<https://www.tensorflow.org/lite/tutorials/model_maker_image_classification?hl=ja>

#### tensorboard

Webブラウザを用いてtensorflowの学習状況を可視化するツール。

起動時にtensorflowのlogディレクトリを指定する必要がある。

例

tensorboard --logdir ./log

### トラブルシューティング

#### cudart64\_110.dll not found

NVIDIA GPU Computing Toolkitをインストール。

またはGPU（グラフィックボード）を使用する。

#### Import "tensorflow.contrib" could not be resolved

このあたりが原因

・tensorflow.contribはtensorflow1.n系でのみ使用

・python3.7以降ではtensorflow1.n系は非対応。

## PyTorch

### 概要

主に自然言語処理に使用される。

nvidiaが開発チームに入っている？

### インストール

公式　<https://pytorch.org/>　でOSやマシンに合わせたインストール方法の判別ができる。

例：Linux でGPU無し（CPU

pip3 install torch torchvision torchaudio --extra-index-url https://download.pytorch.org/whl/cpu

例：Windows でGPU無し（CPU

pip3 install torch torchvision torchaudio

### チュートリアル（推論）

#### 静止画（Pillow）

##### 前準備

.pthファイルを用意する

（但し、必要な時にはtorchが自動的にダウンロードする）

|  |  |  |
| --- | --- | --- |
| モデル | URL | ラベル |
| SSD MobileNetV3 | https://download.pytorch.org/models/ssdlite320\_mobilenet\_v3\_large\_coco-a79551df.pth |  |
|  | https://download.pytorch.org/models/resnet50-0676ba61.pth |  |

補足：ラベルの文字列は基本的にはmodels.classesプロパティで取得できるらしいが、モデルによっては別途入手する必要がある

配置場所

|  |  |
| --- | --- |
| OS | 配置場所 |
| Ubuntu | /home/[ユーザー名]/.cache/torch/hub/checkpoints/ |
| windows | C:\Users\[ユーザー名]\.cache\torch\hub\checkpoints\ |

##### ソースコード

import os

import torch

import torchvision

from PIL import Image, ImageDraw

# Faster R-CNN ResNet50

# model = torchvision.models.resnet50(weights=torchvision.models.ResNet50\_Weights.IMAGENET1K\_V1)

# SSD MobileNetV3

model = torchvision.models.detection.ssdlite320\_mobilenet\_v3\_large(weights=torchvision.models.detection.SSDLite320\_MobileNet\_V3\_Large\_Weights.DEFAULT)

# YOLOv3

model.eval()

# 画像を読み込む

img\_path = os.path.join('images', 'grace\_hopper.jpg')

img = Image.open(img\_path).convert('RGB')

print('width:%s height:%s' % (img.width, img.height))

transform = torchvision.transforms.Compose([torchvision.transforms.ToTensor()])

# 画像データをテンソルに変換 -> (1, 4) のテンソルに変換

img\_tensor = transform(img).unsqueeze(0)

# 推論処理 ※モデルに依って結果が変わってくるらしいので、モデル別に確認が必要

predictions = model(img\_tensor)

# print(predictions[0], type(predictions[0]))

labels = predictions[0]['labels'].tolist()

scores = predictions[0]['scores'].tolist()

boxes = predictions[0]['boxes'].tolist()

draw = ImageDraw.Draw(img)

for label, score, box in zip(labels, scores, boxes):

if score > 0.1:

x, y, w, h = box

# バウンディングボックスの描写

draw.rectangle((x, y, w, h))

img.save('images/result.jpg', quality=95)

参考：<https://qiita.com/grapefruit1030/items/0364c38cdfac892273f3>

公式　ssdlite：[https://github.com/pytorch/vision/blob/29418e34a94e2c43f861a321265f7f21035e7b19/torchvision/models/detection/ssdlite.py#L234](https://github.com/pytorch/vision/blob/29418e34a94e2c43f861a321265f7f21035e7b19/torchvision/models/detection/ssdlite.py" \l "L234)

#### 領域（Segmentation）

公式？：<https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html>

### チュートリアル（学習）

#### モデル作成

import torch.nn as nn

import torch.nn.functional as F

class CNN(nn.Module):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.conv1 = nn.Conv2d(in\_channels=1, # 入力チャネル数

out\_channels=32, # 出力チャネル数

kernel\_size=3, # フィルターサイズ

padding=(1,1), # パディングを行う

padding\_mode='zeros') # ゼロでパディング

self.pool1 = nn.MaxPool2d((2, 2)) # プーリング2x2

self.conv2 = nn.Conv2d(in\_channels=32, # 入力チャネル数

out\_channels=64, # 出力チャネル数

kernel\_size=3, # フィルターサイズ

padding=(1,1), # パディングを行う

padding\_mode='zeros') # ゼロでパディング

self.pool2 = nn.MaxPool2d((2, 2)) # プーリング2x2

self.conv2\_drop = nn.Dropout2d(0.5) # 畳み込みのドロップアウト

self.fc1 = nn.Linear(in\_features=7\*7\*64, # 入力はフラット化後のサイズ

out\_features=128) # ニューロン数

self.fc\_drop = nn.Dropout(0.5) # ドロップアウト

self.fc2 = nn.Linear(in\_features=128, # 入力のサイズは前層のニューロン数

out\_features=10) # ニューロン数はクラス数と同数

def forward(self, x):

x = self.pool1(F.relu(self.conv1(x))) # 畳み込み層1->ReLU適用->2x2のプーリング(28, 28, 1) -> (14, 14, 64)

# 畳み込み層2の出力->ReLU適用->2x2のプーリング

# (バッチサイズ, 14, 14, 64) -> (バッチサイズ, 7, 7, 64)

x = self.pool2(F.relu(self.conv2(x)))

x = self.conv2\_drop(x) # ドロップアウト

x = x.view(-1, 7 \* 7 \* 64) # (バッチサイズ,7,7,64)->(バッチサイズ,3136)

# 全結合層1の出力->ReLU適用

# (バッチサイズ,3136)->出力(バッチサイズ, 128)

x = F.relu(self.fc1(x))

x = self.fc\_drop(x) # ドロップアウト

x = self.fc2(x) # 全結合層2(バッチサイズ, 128)->(バッチサイズ, 10)

return x

#### 資料

<https://pytorch.org/tutorials/beginner/basics/quickstart_tutorial.html>

### 具体例

#### 位置判定

直線的な判定の場合は比較的簡単。大きく２通りの方法がある：

filterメソッドを用いる場合

# 推論

results = model(source)

# 自作クラスにまとめる

ret\_list = [DetectionResult(

xywhn=BoxInfo(

x = float(box.xywhn[0][0]),

y = float(box.xywhn[0][1]),

width = float(box.xywhn[0][2]),

height = float(box.xywhn[0][3]),

),

) for box in boxes]

# エリアごとの検出結果

left\_side = list(filter(lambda arg: arg.xywhn.x < 0.5, ret\_list))

right\_side = list(filter(lambda arg: arg.xywhn.x >= 0.5, ret\_list))

torch.Tensorのブロードキャスト機能を用いる場合

import torch

# 推論

results = model(source)

# 結果の取得

boxes = results[0].boxes

# エリアごとの検出結果

if len(boxes) > 0:

boxes\_left\_side = boxes[torch.where(boxes.xywhn[0][0] < 0.5)] # 補足：xywhnはx, y, w, h の順番

boxes\_right\_side = boxes[torch.where(boxes.xywhn[0][0] >= 0.5)]

比較

|  |  |  |
| --- | --- | --- |
|  | 長所 | 短所 |
| クラスfilter式 | ・ソースコードが分かりやすい | ・クラスのListにする処理が冗長 |
| Tensor式 | ・ブロードキャスト機能が使える | ・ソースコードが読みにくい |

補足

演算速度に差が出ると思ったが、ほとんど無かった。2023-09-08

GPUあると結果も違うかも。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 平均[mSec] | 中央[mSec] | 最小[mSec] | 最大 [mSec] |  |  |
| クラスfilter式 | 0.331807478 | 0.32434755 | 0.2700062 | 0.4001943 |  |  |
| Tensor式 | 0.320294644 | 0.3045282 | 0.2717485 | 0.4126078 |  |  |

#### GradCAM

YOLOv8はGradCAMには対応していないかもしれない。

但し、別の方法でヒートマップを表示する事はできるらしい

<https://github.com/ultralytics/ultralytics/issues/2020>

資料

pytorch向きのpipライブラリが2つあるらしい

grad-cam

こちらの方がはるかに人気

pip install grad-cam

pypi：<https://pypi.org/project/grad-cam/>

Github：<https://github.com/jacobgil/pytorch-grad-cam>

pytorch-gradcam

pip install pytorch-gradcam

pypi：<https://pypi.org/project/pytorch-gradcam/>

### レファレンス（torch）

#### tensor

int\_tensor = torch.tensor(float\_tensor, dtype=torch.int32) # 型変換

参考

チートシート：<https://qiita.com/awawaInu/items/e173acded17a142e6d02>

#### where

要素のindex番号が返る点に注意

構文：

torch.where(Tensor condition)

torch.where(Tensor condition, Tensor input, Tensor other, \*, Tensor out)

torch.where(Tensor condition, Number self, Tensor other)

torch.where(Tensor condition, Tensor input, Number other)

torch.where(Tensor condition, Number self, Number other)

補足：input, otherはブロードキャスト(numpy)可能でなければならない

具体例

boxes = boxes[torch.where(boxes.cls == 0.0)] # class\_id==0のものだけ抽出

公式：<https://pytorch.org/docs/stable/generated/torch.where.html>

補足

#### torch.utils.tensorboard

公式：<https://pytorch.org/docs/stable/tensorboard.html>

##### SummaryWriter

### レファレンス（torchvision）

#### torchvision.ops

公式：<https://pytorch.org/vision/0.8/ops.html>

##### nms

重複するバウンディングボックスを削除する

書式：torchvision.ops.nms(boxes: Tensor, scores: Tensor, iou\_threshold: float) → Tensor[SOURCE]

例

# Non-Maximum Suppression で余計な box を削除

boxes = torchvision.ops.nms(boxes, boxes.conf, iou\_threshold=0.5)

YOLO(ultralytics)の場合 2023-08-22 やり方が合ってるかの確信はない

# Non-Maximum Suppression で余計な box を削除

boxes\_tensor = torch.tensor(boxes.xyxy)

confs\_tensor = torch.tensor(boxes.conf)

boxes\_nms = torchvision.ops.nms(boxes\_tensor, confs\_tensor, iou\_threshold=0.5)

boxes = boxes[boxes\_nms]

公式：<https://pytorch.org/vision/main/generated/torchvision.ops.nms.html>

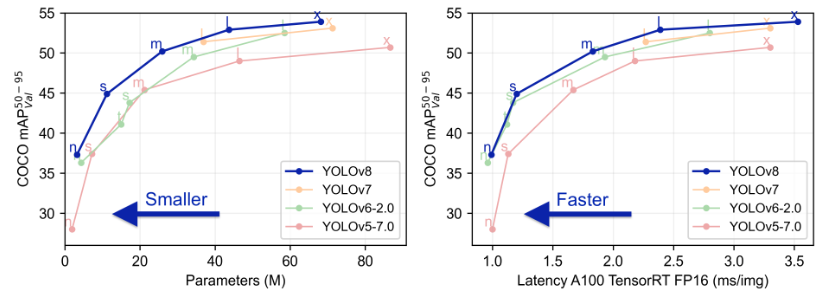
## YOLO

### 概要

torchで使う事が多いが、tensorflowでも使える（らしい）

n,s,m,l,xの5種類あり、後ろにいくほど処理が重く、精度が高くなる。

処理速度と精度の表



情報元：<https://github.com/ultralytics/ultralytics>

補足

・Roboflowを使うと非常に簡単にデータセットを作成できるらしい（要アカウント登録）

<https://roboflow.com/model/yolov8>

[ terminal ]

pip install roboflow

[ python ]

import roboflow

### モデルについて

|  |  |  |  |
| --- | --- | --- | --- |
| モデル | 入手方法 | 開発元 | 参考 |
| yolov5 | UltralyticsのGithub | Ultralytics |  |
| yolov6 | Githubから |  | <https://github.com/meituan/YOLOv6/releases/download/0.3.0/yolov6n.pt>  <https://github.com/meituan/YOLOv6/releases/download/0.3.0/yolov6s.pt> |
| yolov7 |  | WongKinYiu？ | <https://github.com/WongKinYiu/yolov7> |
| yolov8 | UltralyticsのGithub | Ultralytics |  |

それぞれDetection, Segmentation, Classification, Pose が存在する

|  |  |  |  |
| --- | --- | --- | --- |
| 名前 | 概要 | モデルファイル名 | 備考 |
| Detection | 物体検出 | yolov8n | <https://docs.ultralytics.com/tasks/detect/> |
| Segmentation | 領域検出 | yolov8n-seg | <https://docs.ultralytics.com/tasks/segment/> |
| Classification | 画像分類 | yolov8n-cls | <https://docs.ultralytics.com/tasks/classify/> |
| Pose | 姿勢検出 | yolov8n-pose.pt | <https://docs.ultralytics.com/tasks/pose/> |
|  |  |  |  |

### チュートリアル

#### インストールと推論のテスト

環境構築

[ powershell]

# 作業フォルダと必要なファイルの作成（ファイルエキスプローラーでやっても良い）

cd d:\

mkdir yolo\_test

cd yolo\_test

New-Item detect.py

# pythonのバージョン確認

python -V # ver3.7以上が必要

# pythonの仮想環境を作成してアクティベート（任意だが強く推奨）

python -m venv venv # 環境によるが10～20秒くらいかかる

venv\Scripts\activate # 仮想環境を有効化

# おまけ

(venv) PS D:\yolo\_test > deactivate # 仮想環境を無効にしたい場合

rm -r venv\ # 仮想環境を作り直すなどの理由で削除したい場合

補足

ここから先はVSCodeを使った方がやりやすい。

VSCodeを開くFile → OpenFolder → D:\yolo\_testを選択 → フォルダを開く

Ctrl+@でターミナル（powershell）が開く

ディレクトリ構成

yolo\_test/

+ venv/

+ detect.py

次ページに続く

チュートリアル - インストールと推論のテスト　つづき

インストール

[ terminal ]

python3 -m pip install -U pip # pip自体をアップグレードする

python3 -m pip install Pillow==9.5.0 # Pillowはver9以下でないとultralyticsで例外が発生する

python3 -m pip install ultralytics # YOLOを使う為のライブラリ（torchもインストールされる）

detect.pyを編集（コピペでOK）

[ detect.py ]

from ultralytics import YOLO

source\_bus = 'https://ultralytics.com/images/bus.jpg' # Ultralyticsからダウンロードが可能

source\_zidane = 'https://ultralytics.com/images/zidane.jpg' # これでも良い

# source\_zidane = 'data/images/zidane.jpg' # ローカルにあるファイルを指定する事もできる

model = YOLO('yolov8n') # 推論用モデル。初回はダウンロードされる

model =YOLO('models/yolov8s') # ローカルにある .ptファイルを指定する事もできる

model.predict(source\_bus, save=True) # 推論実行

実行する

[ terminal ]

python3 detect.py

> モデルファイルyolov8n.ptがダウンロードされる

> runs/predict{n} に実行結果の画像が保存される

補足：実行後のディレクトリ構成

yolo\_test\

+ runs/detect/

+ predict 初回の推論結果が保存されるディレクトリ

+ predict2 ２回目の推論結果が保存される

+ venv\

+ detect.py

+ bus.jpg 今回ダウンロードされたもの

+ yolov8n.pt 今回ダウンロードされたもの

公式チュートリアル：<https://docs.ultralytics.com/quickstart/>

#### 推論結果の詳細

detect.pyを以下のように変更

[ detect.py ]

from ultralytics import YOLO

source\_bus = 'bus.jpg' # 先ほどダウンロードしたファイルを利用する

model = YOLO('yolov8n', conf=0.2) # スコアが0.2以上の物体に限定する

names = model.names

# 推論結果

results = model(source\_bus)

boxes = results[0].boxes

classess = results[0].boxes.cls

scores = results[0].boxes.conf

# 検出結果を１つずつループで確認する

for index, box in enumerate(boxes):

print('=' \*10, f'box{index}','=' \*10,)

print('xyxy:%s\nxywh:%s\nxyxyn:%s\nxywhn:%s' %

(box.xyxy, box.xywh, box.xyxyn, box.xywhn))

print('class\_id:%s\tlabel:%s\tscore:%s' %

(box.cls, names[int(box.cls)], box.conf))

このような値が出力される

========== box0 ==========

xyxy:tensor([[ 17.2858, 230.5922, 801.5183, 768.4058]]) # 何故かListのListになっている事が分かる

xywh:tensor([[409.4020, 499.4990, 784.2325, 537.8136]])

xyxyn:tensor([[0.0213, 0.2135, 0.9895, 0.7115]])

xywhn:tensor([[0.5054, 0.4625, 0.9682, 0.4980]])

class\_id:tensor([5.]) label:bus score:tensor([0.8705])

補足：Boxesクラスの主要プロパティの概要

|  |  |  |  |
| --- | --- | --- | --- |
| プロパティ名 | 書式 | データ型 | 備考 |
| xyxy | 左上、右下の座標 | torch.float32 | 実際の座標情報 |
| xywh | 中心点と高さ、幅 | torch.float32 | 実際の座標情報 |
| xyxyn | 左上、右下の座標 | torch.float32 | 正規化された数値情報（0.0 ≦ n ≦ 1.0） |
| xywhn | 中心点と高さ、幅 | torch.float32 | 正規化された数値情報（0.0 ≦ n ≦ 1.0） |

公式（Results）：[https://docs.ultralytics.com/reference/engine/results/#ultralytics.engine.results.Results](https://docs.ultralytics.com/reference/engine/results/" \l "ultralytics.engine.results.Results)

公式（Boxes）：[https://docs.ultralytics.com/reference/engine/results/#ultralytics.engine.results.Boxes](https://docs.ultralytics.com/reference/engine/results/" \l "ultralytics.engine.results.Boxes)

補足：Resultsの構造

モジュール：ultralytics.engine.results.py クラス：Results

results = List[Results(

boxes = [

Boxes (

data = [], # xyxyやxywhなどはこのBoxes.dataの値を基にList[List[]] を返すプロパティ

boxes(), # boxesはBoxes.data を返すプロパティ。今後廃止予定（deprecated）。

… # 但し、2023年9月現在、Boxes.dataにアクセスすると「そんなものは無い！」と怒られる

)]

names

…

)]

#### 学習のデモ

Gitレポジトリをクローン

Webブラウザで<https://github.com/ultralytics/ultralytics>へアクセスし、 > Download ZIP

または、Gitがインストールされていれば、UltralyticsのGithubを適当なディレクトリにクローン

git clone https://github.com/ultralytics/ultralytics

> ultralytics\cfg\datasetsにあるcoco128.yamlをyolo\_testにコピー

ディレクトリ構成（抜粋）

/ yolo\_test

+ venv 仮想環境

+ coco128.yaml データセットの情報（メタデータ）を書いたファイル

+ train.py 学習実行するpythonスクリプト（新規作成）

train.pyを編集

[ train.py]

from ultralytics import YOLO

model = YOLO('yolov8n') # 転移学習の元とするモデル

model.train(data='coco128.yaml', epochs=5) # 学習開始

実行

[ terminal ]

python3 train.py

結果として/runs/detect/train/weightsにbest.pt, last.ptが作成される。（重みデータ）

best 学習中において最も精度が良かったデータ

last 最終のデータ

補足

今回のYAMLファイルにはdownloadにてデータセットのデータがダウンロードされる仕組みになっている為、Webへの接続が必要。また、オリジナルの学習データを作成したい場合は自分でデータセットを作成する必要がある。

転移学習の詳細については後述。

資料

公式：<https://docs.ultralytics.com/modes/train/>

家のPC 40sec/1epoch くらい

#### カメラを用いたリアルタイム推論

ディレクトリ構成（抜粋）

yolo\_cv/

|- models/

| |- yolov8n.pt 今回使用するモデル

|- main.py

import cv2

import torch

from ultralytics import YOLO

cap = cv2.VideoCapture(0)

model = YOLO('yolov8n')

# クラスごとに色分けをする仕組み（任意） B,G,R

colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), ]

try:

while cap.isOpened():

ret, frame = cap.read() # ret が実行結果(1 or 0)，frameが取得された画像

# 推論演算を行う

results = model(frame)

boxes = results[0].boxes

names = results[0].names # model.namesの方が使いやすいが、備忘録として

class\_ids = results[0].boxes.cls # 念の為書くが、この形ではあまり使われない

scores = results[0].boxes.conf # この形で使う事は少ないが、備忘録として

annotated\_frame = results[0].plot() # 結果などが描画された画像

# TOP N とスコアの値によって不要な要素を排除

boxes = boxes[:5]

boxes = boxes[torch.where(boxes.conf > 0.5)]

# バウンディングボックスの描画処理（plot() の画像を用いず自分でやる場合）

for box in boxes:

# 変数の変換

x\_min, y\_min, x\_max, y\_max = (int(i) for i in box.xyxy[0])

cls\_id = int(box.cls) # torch.Tensorという型なので、intに直す

name = names[cls\_id]

score = box.conf # スコア。confidence の意と思われる

# バウンディングボックス、クラス名の描画

color = colors[cls\_id % len(colors)] # バウンディングボックスの色

cv2.rectangle(frame, (x\_min, y\_min), (x\_max, y\_max), color, 1)

cv2.putText(frame, name, (x\_min+10, y\_max-10), cv2.FONT\_HERSHEY\_SIMPLEX, 1, color, 1, cv2.LINE\_AA)

# Qt5を用いて画像を表示

cv2.imshow('camera capture', annotated\_frame)

if cv2.waitKey(100) & 0xFF == ord('q'):

break

except KeyboardInterrupt: # Ctrl+C でも終了できるようにする（任意）

pass

cap.release()

cv2.destroyAllWindows()

補足

自作クラスによるまとめ例

# 検出結果をまとめる

det\_results = [DetectionResult(

box=ResultBox(

x\_min = int(box.xyxy[0][0]),

y\_min = int(box.xyxy[0][1]),

x\_max = int(box.xyxy[0][2]),

y\_max = int(box.xyxy[0][3]),

),

score = box.conf,

class\_id=int(box.cls),

label=names[int(box.cls)],

) for box in boxes]

自作クラス

class ResultBox:

def \_\_init\_\_(

self,

x\_min:int,

y\_min:int,

x\_max:int,

y\_max:int,

):

self.x\_min = x\_min

self.y\_min = y\_min

self.x\_max = x\_max

self.y\_max = y\_max

def \_\_str\_\_(self): # toString() の様な物。printなどで文字列に変換する場合に呼び出される。

return "(%s, %s), (%s, %s)" % (self.x\_min, self.y\_min, self.x\_max, self.y\_max)

class DetectionResult:

'''

検出結果の情報をまとめるクラス

motpy.Detection に倣って作成

'''

def \_\_init\_\_(

self,

box:ResultBox,

score:float,

class\_id:int,

label:str,

tracking\_id:Optional[uuid.uuid4]=None,

):

self.box = box

self.score = score

self.class\_id = class\_id

self.label = label

def \_\_repr\_\_(self):

return f'Detection(box={self.box}, score={self.score:.5f}, label={self.label})'

### 転移学習

#### 準備

学習用データセットのディレクトリ構成にまとめる

ディレクトリ構成

yolo\_train/ ここのディレクトリ名は何でも良い

+ models/ 転移学習のもとになる.ptファイルを格納する

+ train.py 学習を実行するスクリプト

+ datasets/ 学習用データを格納する（後述）

+ mymodel / mymodel.ptというモデルファイルを作成する場合

+ train/ 画像ファイルを格納するディレクトリ

+ images/ 画像ファイルを格納するディレクトリ

+ photo1.jpg 実際に学習で用いる画像ファイル

+ labels/ アノテーション情報を格納するディレクトリ

+ photo1.txt 写真のファイル名と対になる、同名のtxtファイルを用意する

アノテーションの中身

YOLO形式での書式は：クラスID [空白] 物体の中心座標X [空白] Y [空白] 幅 [空白] 高さ

[ photo1.txt ]（例）

0 0.295312 0.389583 0.059375 0.079167 # class x座標 y座標 width height

1 0.503125 0.602083 0.581250 0.558333

補足１：class以外は小数点6桁の小数

補足２： labelImgなどのアノテーションソフトを利用するのが圧倒的に楽（後述）

#### 自動アノテーション（任意）

概要

データセットに必要な画像と物体の座標情報は物体認識処理中に手に入る為、物体認識処理と同時にアノテーションファイルを作成する方法例。

画像の保存

opencvのimwriteメソッドを使う事で保存できる

例

# 画像ファイルの保存パスを生成

dir\_path = os.path.join('datasets','images')

fn = datetime.datetime.now().strftime('%Y%m%d\_%H%M%S')

img\_path = os.path.join(dir\_path, (fn+'.png'))

# 実際に保存

cv2.imwrite(img\_path, frame)

補足

Pillowを用いて保存する方法もある

Pythonによるアノテーションファイルの作成例

from ultralytics import YOLO

import torch

source = 'images/target\_img.jpg' # 対象の画像

results = model(source)

# 推論結果

boxes = results[0].boxes

names = results[0].names

# アノテーションファイルに必要なデータをまとめる　正規化された中心点と高さ、幅が必要なのでxywhnを用いる

annotation\_data\_list = [(

int(box.cls),

box.xywhn[0][0], # ListのListになっている為 [0][0] といった形になる

box.xywhn[0][1],

box.xywhn[0][2],

box.xywhn[0][3],

) for box in boxes]

# アノテーションファイルの作成

with open('bus.txt''', 'w') as f:

for dt in annotation\_data\_list:

f.write(f'{dt[0]} {dt[1]:.6f} {dt[2]:.6f} {dt[3]:.6f} {dt[4]:.6f}\n')

補足

|  |  |  |  |
| --- | --- | --- | --- |
| 名前 | 書式 | データ型 | 備考 |
| xyxy | 左上、右下の座標 | torch.float32 | 実際の座標情報 |
| xywh | 中心点と高さ、幅 | torch.float32 | 実際の座標情報 |
| xyxyn | 左上、右下の座標 | torch.float32 | 正規化された数値情報（0.0 < n < 1.0） |
| xywhn | 中心点と高さ、幅 | torch.float32 | 正規化された数値情報（0.0 < n < 1.0） |

#### YAML作成

##### 設定ファイルの値確認

学習データの情報を記述する.yamlファイルを作成する。

.yamlファイル作成まえに、設定ファイルのある値を確認する。

なお設定ファイル（settings.yaml）の配置場所は以下の通り。

|  |  |
| --- | --- |
| 環境 | 設定ファイルの場所 |
| Windows | C:\Users\[ユーザー名]\AppData\Roaming\Ultralytics\settings.yaml |
| Debian系Linux | /home/[ユーザー名]/.config/Ultralytics/settings.yaml |

この設定ファイルの**「datasets\_dir」の値が「datasets」になっている事を確認**する

⇒ もし違っている場合は修正する必要がある。

変更方法１：設定ファイルを直接変更

[ settings.yaml ]

datasets\_dir: datasets

変更方法２：pythonで変更

[ python ]

from ultralytics import settings

settings['datasets\_dir'] = 'datasets'

settings.update()

補足

学習の際にdatasets\_dirの値が書き換えられてしまう事があるらしい。

その設定のまま次の学習使用とすると

Dataset 'mymodel.yaml' images not found , missing path 'D:\yolo\_train\datasets\mymodel'

といったエラーが発生する事がある。

この問題は上記を参考にdatasets\_dirの値をdatasetsにすると修復できる。

##### 書き方

ディレクトリ構成

yolo\_train/

+ datasets/

+ mymodel.yaml データセットの情報を記述するファイル

+ mymodel /

+ train /

+ images / 学習用の画像ファイルを配置

+ labels / アノテーションファイル（.txt）を配置

[ mymodel.yaml ]（例）

path: mymodel # settings['datasets\_dir'] からの相対パスで記述する

train: train # path からの相対パスで記述する

val: train

# test: train

names: [person, bicycle, car]

補足１

.yamlファイルはキーと値の間に空白が必要

path: mymodel # 〇

path:mymodel # ×

補足２

train、val、testの値の書き方は以下の３通りある：

|  |  |  |
| --- | --- | --- |
|  | 書き方 | 備考 |
| ディレクトリ | path/to/images |  |
| ファイル | path/to/annotation.txt |  |
| List | [ path/to/img1, path/to/img2 ] |  |

その為、クラス名は以下のように指定しても良い（らしい）

names:

0: person

1: bicycle

2: car

ラベル文字列を.yamlファイルに含まず、別ファイルで指定する方法があるらしい？

⇒ 2023-08-31上手く行かなかった

[ classes.txt ]

person

bicycle

car

##### train, val, testを分ける場合

学習データはtrain, val(validation), testに分けた方がより良い学習モデルが作成できる。

それぞれのデータは一般的には以下のように使われる：

|  |  |
| --- | --- |
| 項目 | 概要 |
| train | 学習の為に用いられるデータ |
| validation  検証データ | ハイパーパラメータが適切であるか確認する為のデータ。学習中に使う。  訓練データを使ってはいけない。（過適合が起こる） |
| test  テストデータ | 学習が完全に終わった後、モデルの評価を行う。 |

ディレクトリ構成

yolo\_test/

+ datasets/

+ mymodel.yaml データセットの情報をまとめたファイル

+ mymodel /

+ classes.txt クラス名が記述してあるファイル

+ train /

+ images /

+ labels /

+ val /

+ images /

+ labels /

+ test /

+ images /

+ labels /

[ mymodel.yaml ]（例）

path: mymodel # settings['datasets\_dir'] からの相対パスで記述する

train: train # path からの相対パスで記述する

val: val

test: test

names: [person, bicycle, car]

なお、データ数の比率の目安は：

testを使わない場合 train : val = 7 : 3

testを使う場合 train : val: test = 6 : 2 : 2

#### 学習実行

##### 学習スクリプト

from ultralytics import YOLO, setting

# 転移学習元のモデルを指定する

model = YOLO('yolov8n.pt')

# 学習する

model.train(data='mymodel.yaml', epochs=100, batch=10, imgsz=640)

パラメータ（抜粋）

|  |  |  |  |
| --- | --- | --- | --- |
| 名前 | データ型 | 概要 | 初期値 |
| lr0 | float | 初期の学習率  学習の最初は大きく、最後の方は小さくなる事が理想 | 0.01 |
| batch | int | 学習で使う「１塊」のデータ数。重要パラメータ。  -1で自動 | 16 |
| epochs | int | ミニバッチ学習を繰り返す回数 | 100 |
| weight\_decay | float | 過学習を防ぐための値 | 0.0005 |
| optimizer | str | SGD, Adam, Adamax, AdamW, NAdam, RAdam, RMSProp, auto | 'auto' |
| patience | int | 精度の向上がない時に、学習中止を見送るエポック回数 | 50 |

公式：<https://docs.ultralytics.com/modes/train/>

##### 学習結果（results.csv）

結果はruns/train{n}/results.csv に出力される

なお、学習中にも標準出力に表示される。

説明

train またはval

box\_loss 物体検出モデルが予測するボックスと、実際のボックスの位置とサイズとの間の誤差

cls\_loss 物体検出モデルが予測する物体のクラスと実際との誤差

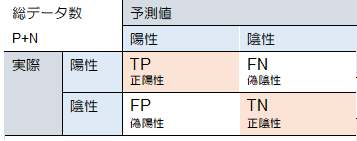
dfl\_loss 物体の形状変化(deformation)を考慮した際の誤差？

⇒ いずれもだんだん小さくなるのが理想

metrics

precision(B) 予測精度：陽性予測の総数の内、実際に陽性であった割合 式：

recall(B) 再現率：実際の陽性データのうち、陽性と予測できた割合 式：

補足

TP：正陽性（陽性と予想し、実際に陽性）

TN：正陰性（陰性と予想し、実際に陰性）

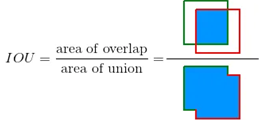
FP：疑陽性（陽性の予想に対し、実際には陰性）

FN：疑陰性（陰性の予想に対し、実際には陽性）

mAP50(B) IOUが50以上であれば正解、とみなした時の精度

mAP50-95(B) IOU=50~95であれば正解、とみなした時の精度

補足

IOU：予想ボックスと実際ボックスの精度の指標

= 重複部分÷合算部分 で求める（右図参照）

⇒ いずれも高い状態で終わるのが理想

lr

pg0～pg*n* 学習率

その他

Images val(validationの意)ディレクトリに入っている画像の数（？）

Instance valディレクトリ内の画像全てに検出演算を行い、検出された物体数の総数（？）

参考資料

[https://tech.aru-zakki.com/yolov8-train-and-infer#rtoc-6](https://tech.aru-zakki.com/yolov8-train-and-infer" \l "rtoc-6)

##### TensorBoard

学習中１エポックごとに出力される各データを可視化してくれるツール

YOLOの場合はtrain\results.csvの内容を可視化する（らしい）

補足

横軸はエポック数

起動方法

[ terminal ]

# インストール

pip install tensorboard

# 実行

tensorboard --logdir runs

補足：--logdirには学習結果が出力されるディレクトリを指定する

Google Colaboratoryの場合：

このようなコードセルを用意してそのセルを実行するとインラインで表示される

%load\_ext tensorboard

%tensorboard --logdir runs

使い方メモ

画面の自動更新

 > Reload dataにチェック > Reload Period(seconds) で更新頻度を設定（30秒が最小）

### 物体追跡

#### 概要

Ultralytic公式の追跡アルゴリズム

|  |  |  |
| --- | --- | --- |
| アルゴリズム | trackerの設定値 | 備考 |
| BoT-SORT | botsort.yaml | 2022年6月発表のアルゴリズム |
| ByteTrack | bytetrack.yam | 2021年10月発表のアルゴリズム |

情報元：<https://github.com/ultralytics/ultralytics/issues/1429>

#### YOLO標準の方法

ソースコード

import os

from ultralytics import YOLO

model = YOLO('yolov8n')

source = os.path.join('images', 'sample.mp4')

model.track(

source = source,

show = True,

tracker="bytetrack.yaml",

)

#### OpenCVを用いる方法

import os

import cv2

from ultralytics import YOLO, settings

model = YOLO('yolov8n')

cap = cv2.VideoCapture('sample.mp4')

# 動画の総フレーム数を取得

frames\_count = int(cap.get(cv2.CAP\_PROP\_FRAME\_COUNT))

for \_ in range(frames\_count):

ret, frame = cap.read()

if ret:

results = model.track(

source = frame,

conf= 0.3,

iou = 0.5,

show = True,

)

annotated\_frame = results[0].plot() # 結果などが描画された画像

# opencvのimshowメソッドを使っても良い

# cv2.imshow('video capture', annotated\_frame)

if cv2.waitKey(10) & 0xFF == ord('q'): # キーボードで q を押したら終了する

break

cap.release()

なお、以下のようにする事でboxの中身を確認する事ができる

# 検出結果を１つずつループで確認する

for index, box in enumerate(boxes):

print('=' \*10, f'box{index}','=' \*10,)

print(box[0].id)

print(box)

boxのフィールドまとめ（物体検出に特化したもののみ）

|  |  |  |
| --- | --- | --- |
|  | データ型 | 概要 |
| id | torch.float32 | トラッキングID |
| is\_track | bool |  |

公式：[https://docs.ultralytics.com/modes/track/#persisting-tracks-loop](https://docs.ultralytics.com/modes/track/" \l "persisting-tracks-loop)

### 共通事項

#### 写真のダウンロード

|  |  |
| --- | --- |
| bus | https://ultralytics.com/images/bus.jpg |
| zidane | https://ultralytics.com/images/zidane.jpg |

### リファレンス

#### mymode.yaml

nc: 3 # クラス数（任意）

download # 学習直前にダウンロードするファイル

以下のようにする事で、学習前にpythonスクリプトを実行する事もできる（らしい）

download:

公式：[https://docs.ultralytics.com/datasets/detect/argoverse/#dataset-yaml](https://docs.ultralytics.com/datasets/detect/argoverse/" \l "dataset-yaml)

公式： <https://docs.ultralytics.com/datasets/detect/>

#### データセット

公式：<https://docs.ultralytics.com/datasets/>

#### Boxes

正式名：ultralytics.engine.results.Boxes

|  |  |  |  |
| --- | --- | --- | --- |
| 名前 | 書式 | データ型 | 備考 |
| xyxy | 左上、右下の座標 | torch.float32 |  |
| xywh | 中心点と高さ、幅 | torch.float32 |  |
| xyxyn | 左上、右下の座標 | torch.float32 | 正規化された数値情報（0.0 < n < 1.0） |
| xywhn | 中心点と高さ、幅 | torch.float32 | 正規化された数値情報（0.0 < n < 1.0） |

補足：データ型はtorch.Tensor(dtype=\*)

#### model

##### YOLO（コンストラクタ？）

model = YOLO('yolov8n.yaml') # YAMLからモデルをビルドする場合

model = YOLO('yolov8n.yaml').load('yolov8n.pt') # YAMLからビルドして重みを更新する場合

公式：<https://docs.ultralytics.com/reference/engine/model/>

model（メソッド）

from ultralytics import YOLO

source\_bus = 'https://ultralytics.com/images/bus.jpg' # Ultralyticsからダウンロードが可能

model = YOLO('yolov8n') # 推論用モデル。初回はダウンロードされる

model(source\_bus) # 推論実行

なお第一引数sourceには、静止画、動画、opencvのframe、Web上の動画URL　などを指定できる

パラメータ

|  |  |  |  |
| --- | --- | --- | --- |
| 名前 | データ型 | 概要 | 初期値 |
| source |  | 検出対象 |  |
| conf | float | 検出しきい値 | 0.25 |
| line\_thickness |  |  |  |

資料

公式：[https://docs.ultralytics.com/reference/engine/model/#ultralytics.engine.model.Model](https://docs.ultralytics.com/reference/engine/model/" \l "ultralytics.engine.model.Model)

結果（type=result）: <https://docs.ultralytics.com/reference/engine/results/>

Box：[https://docs.ultralytics.com/reference/engine/results/#ultralytics.engine.results.Boxes](https://docs.ultralytics.com/reference/engine/results/" \l "ultralytics.engine.results.Boxes)

参考

パラメータ：[https://tech.aru-zakki.com/yolov8-train-and-infer#rtoc-8](https://tech.aru-zakki.com/yolov8-train-and-infer" \l "rtoc-8)

##### exportメソッド

tensorflowなどの他のモデルに変換する

from ultralytics import YOLO

# Load a model

model = YOLO('yolov8n.pt') # load an official model

# Export the model

model.export(format='onnx')

対応形式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Format | 引数の値 | Model | Meta  data | 引数 |
| PyTorch | - | yolov8n.pt | ✅ | - |
| TorchScript | torchscript | yolov8n.torchscript | ✅ | imgsz, optimize |
| ONNX | onnx | yolov8n.onnx | ✅ | imgsz, half, dynamic, simplify, opset |
| OpenVINO | openvino | yolov8n\_openvino\_model/ | ✅ | imgsz, half |
| TensorRT | engine | yolov8n.engine | ✅ | imgsz, half, dynamic, simplify, workspace |
| CoreML | coreml | yolov8n.mlpackage | ✅ | imgsz, half, int8, nms |
| TF SavedModel | saved\_model | yolov8n\_saved\_model/ | ✅ | imgsz, keras |
| TF GraphDef | pb | yolov8n.pb | ❌ | imgsz |
| TF Lite | tflite | yolov8n.tflite | ✅ | imgsz, half, int8 |
| TF Edge TPU | edgetpu | yolov8n\_edgetpu.tflite | ✅ | imgsz |
| TF.js | tfjs | yolov8n\_web\_model/ | ✅ | imgsz |
| PaddlePaddle | paddle | yolov8n\_paddle\_model/ | ✅ | imgsz |
| ncnn | ncnn | yolov8n\_ncnn\_model/ | ✅ | imgsz, half |

<https://docs.ultralytics.com/modes/export/>

##### predictメソッド

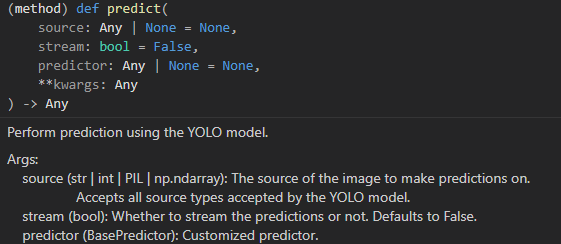
推論した画像内に「どのオブジェクトがどれくらい出現したか」を返す

パラメータ

|  |  |  |  |
| --- | --- | --- | --- |
| 名前 | データ型 | 概要 | 初期値 |
|  |  |  |  |
| conf | float | 検出しきい値 | 0.25 |
| iou | float | NMSの為の閾値  大：余計なボックスが削除されやすくなる  小：重なりが小さいものも報告。ボックスが多くなる | 0.7 |
| imgsz | int | 画像サイズ（スカラー値） | 640 (640,480)の意らしい |
| save | bool | 推論後のファイルを保存するかどうか | False |

公式：<https://docs.ultralytics.com/modes/predict/>

公式（パラメータ）：[https://docs.ultralytics.com/modes/predict/#inference-arguments](https://docs.ultralytics.com/modes/predict/" \l "inference-arguments)



##### trainメソッド

パラメータ（抜粋）

一般的に重要と言われているもの

|  |  |  |  |
| --- | --- | --- | --- |
| 名前 | データ型 | 概要 | 初期値 |
| lr0 | float | 初期の学習率  学習の最初は大きく、最後の方は小さくなる事が理想  関連：cos\_lr | 0.01 |
| lrf | float | 最終的な学習率を決める為の  最終的な学習率 = lr0 \* lrf  関連：cos\_lr | 0.01 |
| cos\_lr | bool | 学習率の変化を緩やかにする  Cosine Annealing LR  関連：lr0，lrf | False |
| batch | int | 学習で使う「１塊」のデータ数。重要パラメータ。  -1で自動 | 16 |
| epochs | int | ミニバッチ学習を繰り返す回数 | 100 |
| weight\_decay | float | 過学習を防ぐための値 | 0.0005 |
| optimizer | str | SGD, Adam, Adamax, AdamW, NAdam, RAdam, RMSProp, auto | 'auto' |
| patience | int | 精度の向上がない時に、学習中止を見送るエポック回数 | 50 |

それ以外

|  |  |  |  |
| --- | --- | --- | --- |
| 名前 | データ型 | 概要 | 初期値 |
| device | int/str | 学習にcuda(gpu)かcpuどちら使うか | None |
| deterministic | bool | deterministicモードを有効にするかどうか。再現性に影響 | True |
| fraction | float | datasetがtrain, valに分割されていないときに、trainデータを使う割合。 例 val=0.6⇒６割がtrain ４割がval | 1.0 |
| imgsz | int | 学習時の画像サイズ（幅？） | 640 |
| pretrained | bool | 学習済みモデルを利用するかどうか | False |
| save | bool | 学習中に予測結果とチェックポイントを保存するか | True |
| workers | int | 学習時につかうスレッドの数 | 8 |
| warmup\_epochs | float | ウォームアップのエポック数 | 3.0 |
| val | bool | 学習時にvalidateを行うかどうか | True |
| verbose | bool | 詳細情報の表示 | False |

公式：<https://docs.ultralytics.com/modes/train/>

参考：[https://tech.aru-zakki.com/yolov8-train-and-infer#rtoc-6](https://tech.aru-zakki.com/yolov8-train-and-infer" \l "rtoc-6)

##### trackメソッド

パラメータ

|  |  |  |  |
| --- | --- | --- | --- |
| 名前 | データ型 | 概要 | 初期値 |
| source |  |  |  |
| show | bool |  |  |
| tracker |  |  |  |
| persist | bool | Trueにするとidがどんどん切り替わっていく | False |

conf, iou, showはpredeictと同じ

戻り値：List[ultralytics.engine.results.Results]

公式：<https://docs.ultralytics.com/modes/track/>

#### NMS

# Non-Maximum Suppression で余計な box を削除

# 2023-08-22 やり方が合ってるかの確信はない ⇒ 2023-08-22 効果なかった

# boxes\_tensor = torch.tensor(boxes.xyxy)

# confs\_tensor = torch.tensor(boxes.conf)

# boxes\_nms = torchvision.ops.nms(boxes\_tensor, confs\_tensor, iou\_threshold=0.5)

# boxes = boxes[boxes\_nms]

#### settings

公式：[https://docs.ultralytics.com/quickstart/#ultralytics-settings](https://docs.ultralytics.com/quickstart/" \l "ultralytics-settings)

#### yolo.data.augment

##### LetterBox

#### yolo.utils.plotting

##### Annotator

##### colors

#### yolo.utils

##### ops

non\_max\_suppression

公式：[https://docs.ultralytics.com/reference/utils/ops/#ultralytics.utils.ops.non\_max\_suppression](https://docs.ultralytics.com/reference/utils/ops/" \l "ultralytics.utils.ops.non_max_suppression)

#### yaml\_save

公式：[https://docs.ultralytics.com/reference/utils/\_\_init\_\_/?h=yaml#ultralytics.utils.yaml\_save](https://docs.ultralytics.com/reference/utils/__init__/?h=yaml" \l "ultralytics.utils.yaml_save)

#### yaml\_load

公式：[https://docs.ultralytics.com/reference/utils/\_\_init\_\_/?h=yaml#ultralytics.utils.yaml\_load](https://docs.ultralytics.com/reference/utils/__init__/?h=yaml" \l "ultralytics.utils.yaml_load)

### 資料

YOLOv3

arxiv：<https://arxiv.org/abs/1804.02767>

公式：<https://pjreddie.com/darknet/yolo/>

Github：

Tiny YOLOv3

軽量なYOLO

YOLOv5

arxiv：

公式：

Github： https://github.com/ultralytics/yolov5

インストール

git clone https://github.com/ultralytics/yolov5.git

cd yolov5/

pip3 install -r requirements.txt

YOLOv6

arxiv：<https://arxiv.org/abs/2209.02976>

公式：

Github：<https://github.com/meituan/YOLOv6>

YOLOv7

概要

arxiv：<https://arxiv.org/abs/2103.14259>

公式：<https://pjreddie.com/darknet/yolo/>

Github：<https://github.com/WongKinYiu/yolov7>

YOLOv8

概要

2023年1月上旬に公開された

arxiv：<https://arxiv.org/abs/2103.14259>

公式：<https://pjreddie.com/darknet/yolo/>

Github：<https://github.com/ultralytics/ultralytics>

install

pip install torch

pip install ultralytics

## OpenCV

### 概要

カスケード分類器という，NG画像の辞書のようなファイルを作成し，それに基づいて不正解を判断してくれるアプリ。

元々はC言語，現在ではC++で開発されており，Gitにソースコードも上がっている。

<https://opencv.org/>

C#（.NET）やpython用のラッパーがあるので，C++が理解できなくても利用する事は可能。

但し， C++ が読めると各種ドキュメントのより深い理解に繋がる。

対応言語

公式

C++，Python，Java，Matlab，C

非公式

C#，Go，Processing，Lua，Ruby，PHP，Haskell

対応環境

Windows，Linux，Mac OS，FreeBSD

Android，iOS，ARM Linux

カスケード分配器のファイルは自分で作る事もできるが，割と難しい上，数時間単位の時間がかかる。

人間の顔，目，体などのカスケードファイルが最初から含まれている為，まずはそれを利用する。

また，画像認識だけでなく，カメラ画像のモニタ，カメラから静止画をキャプチャ，画像の加工（モザイクなど），文字認識，モーションセンサなど色々な事が行えるらしい。

用語

弱識別機：単独の弱精度の識別機。これを複数合わせて作る精度の良い識別機を「強識別機」と呼ぶ。

特徴量：対象物の特徴を定量的に表した変数。

その他

GUIはQtを使っているらしい

#### V4.0

cmakeはv3.5以上が必要

### ソフトウェア情報

#### ライセンス

opencv 商用利用可能

opencv\_contrib 商用利用できない機能も含まれる

#### 学習済カスケードファイル

OpenCVにはHaar-like特徴で学習済のカスケードファイルがある。

以下はダウンロードと利用方法

・このレポジトリをcloneする。<https://github.com/opencv/opencv>

・cloneしたレポジトリにopencv/data/haarcascades ディレクトリがある事を確認する。

・上記のopencv/data/haarcascadesディレクトリをプロジェクトのベースディレクトリにコピー。

|  |  |  |
| --- | --- | --- |
| ファイル名 | 概要 | 備考 |
| haarcascade\_eye.xml | 目 |  |
| haarcascade\_eye\_tree\_eyeglasses.xml | 眼鏡 |  |
| haarcascade\_frontalface\_alt.xml | 顔（正面） |  |
| haarcascade\_frontalface\_alt2.xml | 顔（正面） |  |
| haarcascade\_frontalface\_alt\_tree.xml | 顔（正面） |  |
| haarcascade\_frontalface\_default.xml | 顔（正面） |  |
| haarcascade\_fullbody.xml | 全身 |  |
| haarcascade\_lefteye\_2splits.xml | 左目 |  |
| haarcascade\_lowerbody.xml | 下半身 |  |
| haarcascade\_profileface.xml | 顔（証明写真） |  |
| haarcascade\_righteye\_2splits.xml 右目 |  |  |
| haarcascade\_smile.xml | 笑顔 |  |
| haarcascade\_upperbody.xml | 上半身 |  |
| haarcascade\_frontalcatface.xml | 猫の顔（正面） |  |
| haarcascade\_frontalcatface\_extended.xml | 猫の顔（正面） |  |
| haarcascade\_licence\_plate\_rus\_16stages.xml | ロシアのナンバープレート（全体） |  |
| haarcascade\_russian\_plate\_number.xml | ロシアのナンバープレート（数字） |  |

#### haarcascade（カスケード分類器）の作成

##### １．実行ファイルの用意

Windows用のOpenCVをダウンロードするとカスケード分類器を作成するアプリもついてくる。

但し，バージョンはOpenCV3以前のものとする。

理由：バージョン４以降にはこの３つの実行ファイルが含まれていない為：

opencv\_createsamples.exe；opencv\_traincascade.exe；opencv\_world346.dll

（参照）<https://qiita.com/takanorimutoh/items/5bd88f3d17239a147581>

（公式）<https://opencv.org/opencv-3-4/> →インストーラー

（Git）<https://github.com/opencv/opencv/tree/3.4/apps> →DLしてコンパイル

ディレクトリ構成 ※この通りでなくても良い

cascade\_make

├ bg 背景画像を入れる。（結局NG画像を使った方が楽　20.04.13まだ実験段階）

├ bin ダウンロードしたOpenCVの実行ファイルやdllを入れる。

├ xml 今から作成するカスケード分類器が保存される

├ sample 正解画像と背景画像を組み合わせたサンプル画像が保存される。

├ neg NG画像を入れる。

├ pos 正解画像を入れる。

├ vec カスケード分類器作成に必要な「ベクトルファイル」が保存される。

カスケード作成全体の注意点

・ファイル名に空白，全角文字は使えない。しかもダブルクォーテーションも使えない。

・ディレクトリを示す文字は 「/」でも「\」でも可能。

・BOMありのテキストファイルは使用できない。（⇒対応できないならANSIを使う）

##### ２．サンプル画像の作成

先に注意点。ベクトルサイズは30×30～60×60 くらいまで。100×100以上だと膨大な時間がかかってしまう。20×20以下だと精度が出ない。正解画像の大きさは，このcreatesampleで拡大縮小するので問題にならない。

正解画像と背景画像を用意する。

正解画像条件

・対象物が写真のほとんどを占め，「ノイズ」の少ない，質の良い「教師データ」である事。

・対象物以外の背景色を加工できる場合は精度が若干良くなるが，必須ではない。

背景画像条件

・正解画像より大きいサイズ，最低でも倍くらいはないとエラーが起こる。サイズはばらばらで良い。

・正解画像が含まれていないもの。（まったく無関係ない風景画や動物など）。

・白黒，カラーは問わない。

・背景画像の**相対**ファイル名を記した.txtファイルを用意。ここではbg\_list.txtとし，bgフォルダにいれる。

cmdで以下を実行：

opencv\_createsamples.exe -img pos/pos\_img\_01.jpg -bg bg¥bg\_list.txt -info sample¥info.txt -num 50 -maxxangle 0.5 -maxyangle 0.5 -maxzangle 0.3 -bgcolor 255 -bgthresh 8 -w 48 -h 48

|  |  |  |
| --- | --- | --- |
| オプション | 意味 | 備考 |
| -img | 正解画像を相対パスで指定。 |  |
| -bg | 背景画像の相対パスを保持する.txtを指定。 |  |
| -info | ベクトルファイル作成で使用するファイル。 |  |
| -num | 作成するサンプルの数。背景ファイル数以下にする。 | def. 1000 |
| -w | ベクトルのサイズ(-w：width，-h：height)  元画像よりこのサイズが小さければ縮小して貼り付ける事になり，大きければ拡大となる。 | def. 24　　50～100くらいが良い |
| -h | def. 24　　50～100くらいが良い |
| -maxidev | 最大明度 | def. 40 |
| -maxxangle | X方向の最大回転角度。但し値がラジアン。  他にYとZがある。 | def. x1.1 y1.1 z0.5 |
|  |  |  |
| -maxscale |  | def. 1.0 |
| -bgcolor | 正解画像の背景色。但しグレースケール。 | def.0　　0が黒。255が白。 |
| -bgthresh | ~~正解画像の背景の透過度~~　　やはり透過率かも  背景色とみなされる範囲 | ~~0が100%透過~~  bgcolor±bgthresh の範囲が  背景色とみなされる def.80 |
| -show | 処理が終わった時に変形させた画像が表示される | Enter 次へ　　Esc 終了 |

細かい設定項目

|  |  |  |
| --- | --- | --- |
| オプション | 意味 | 備考 |
| -inv | 色の反転。値は無く，記述する事で適用。 | 未使用 20.04.13 |
| -randinv | ランダムに反転。値は無く，記述する事で適用。 | 未使用 20.04.13 |
| -rngseed | 乱数発生のシード値。-maxxangle などで使う。 | つまり疑似乱数。 def. 12345 |

備考：

-w，-h は正解画像の大きさではなく，検出とみなす最小サイズ。ここで仮に100×100を選択すると，それ未満のサイズ，例えば80×80の対象物は検出されなくなってしまう。

当然解像度は高い方が検出精度は良い。

実用的な精度を得るための必要サンプル数は，正解画像7000枚，NG画像3000枚と言われている。

サンプル画像の作成に失敗すると，info.txt の中身が空になる。念のため中身を確認する事。

##### ３．ベクトルファイルの作成

cmdで以下を実行：

opencv\_createsamples.exe

-info ..¥Positives¥positives.txt

-bg ..¥Negatives¥negatives.txt

-vec .¥vector¥croppkamon.vec

-num 7680 -w 48 -h 48

|  |  |  |
| --- | --- | --- |
| オプション | 意味 | 備考 |
| -info | createsampleで自動的に作られたinfoファイルを用いる。 | .infoファイルが複数ある時は，当然それらを統合する必要がある。 |
| -bg | NG画像の**絶対**パス書かれたリスト。 | 背景の時は相対だったが，こちらは絶対パス |
| -vec | 今から作成する .vecファイル。 |  |
| -num | 作成するサンプルの数。 | def.1000 |
| w, h |  | ここで指定したのと同じサイズを分類器作成の時に指定する必要がある。 |

ここで作成した２つ以上の.vecファイルを合成する事が可能らしい。

<https://github.com/wulfebw/mergevec>

##### ４．分類器の作成

作成されたベクトルファイルからカスケード分配器を作成する。

（例）

opencv\_traincascade.exe -data cascade -vec vec/positive.vec -bg neg/neglist.txt

|  |  |  |
| --- | --- | --- |
| オプション | 意味 | 備考 |
| -data | カスケードファイルを保存するディレクトリ | 必須パラメータ |
| -vec | ベクトルファイル保存場所 |
| -bg | ネガティブ画像のリストが記述されたテキストファイル |
| -numPos | 使用する正解画像の数  正解画像として妥当ではない，と弾かれる事があるので，vecファイルで準備した数の8～9割くらいにしておく。※ | def.2000  ＝vec-fileのサンプル数ではない！ |
| -numNeg | 使用する正解画像の数 | def.1000 |
| -w | ベクトルファイルの時と同じ。幅と高さ。  **正解ベクトルファイルと同じにする必要がある。** | def.24 |
| -h | def.24 |
| -featureType | 特徴量の見つけ方を指定  値：HAAR, LBP, HOG | def. HAAR |
| -bt | boost分類器のタイプ  値：DAB, RAB, LB,GAB |  |
|  |  |  |

※具体的には numPose +（numStages-1）\*（1-minHitRate）\* numPose + S

Sはvecファイルからスキップされた全てのサンプル数。つまりminHitRateによっても必要な量は変わってくる。

HAAR-like特微量 明暗差の激しい，立体的な物が得意。人間の顔など。

明暗パターンは，エッジ４通り，線形８通り，中心２通りしかなく，そのパターン表を作って比較する。

学習時間が桁違いに多く必要。精度の高いものを作る際には4日ほどかかる。

（参考）<http://opencv.jp/opencv-2.2/py/objdetect_cascade_classification.html>

LBP特徴 シンプルで平面的な図形のようなものが得意。

認識速度が非常に速い。輝度の分布をヒストグラムにより変換して特徴を判別する。

HOG特微量 平面的かつ輪郭が複雑な場合が得意。２D絵など。

輝度変化の方向をまとめ，それを境界線とみなして特徴を判別する。エッジに着目している。

細かい設定項目

|  |  |  |
| --- | --- | --- |
| オプション | 意味 | 備考 |
| -numStages | ステージ数 | def. 20 |
| -precalcValBufSize | 値バッファメモリサイズ　[ 単位 MB ] | def. 1024　良く分からない。 |
| -precalcIdxBufSize | インデックスバッファサイズ　[ 単位 MB ] | def. 1024　良く分からない。 |
| -numThreads | 処理に使うスレッドの数 | 13が最大 |
| acceptanceRatioBreakValue |  | def. -1 |

boost のパラメータ

|  |  |  |
| --- | --- | --- |
| オプション | 意味 | 備考 |
| maxFalseAlarmRate | 各ステージ許容する最大誤検出率  例えば0.5 → 0.4にするとより精度の良い分類器を期待できる | def.0.5 |
| minHitRate | 各ステージでの最小ヒット率 | def. 0.95 |
| weightTrimRate | 値によりトリミングを行うか指定する | def. 0.95 |
| maxDepth | 弱検出器の最大の深さ | def. 1 |
| maxWeakCount | -maxFalseAlarmRate達成用に必要とされる各ステージでの弱分類器の最大数 | def.100 |

※maxFalseAlarmRateが厳しすぎると，traincascadeの途中でエラーで止まる。

具体的には acceptanceRatio < maxFalseAlarmRate ^ numStage(現在のステージカウント)

また，このmaxFalseAlarmRate ^ numStage はleafFARateと言う変数名らしい

HAARの時のパラメータ

|  |  |  |
| --- | --- | --- |
| オプション | 意味 | 備考 |
| -mode | BASIC  CORE  ALL | HAARの時だけ　def. BASIC |

[OpenCV公式サイトの原文](https://docs.opencv.org/2.4/doc/user_guide/ug_traincascade.html)

[翻訳版](https://iwaki2009.blogspot.com/2013/12/opencv_11.html)

trainCascade実行中の表示について

Precalculation time：の単位はseconds（秒）

HR：Hit Rate 正解と認識された割合。1.0が100%

FA：False Alarm NEG画像内で正解と誤認識された割合。

(v\_falsealarm / numneg)

% of negative samples incorrectly classified as positive.

EXP.ERR AdaBoostアルゴリズムの分類エラー(v\_experr)

consumed 今までのステージで使用したPOS画像のファイル数。

この値が実際のPOS画像ファイル数を超えると…insufficient count of samples エラーが発生する。

##### エラー対応

Train dataset for temp stage can not be filled. Branch training terminated.

実行パスの一部でも全角文字が含まれていると発生。

Bad argument (Can not get new positive sample…

正解画像の枚数不足。ベクトルで作成されたファイル数より（更に）少ない数を指定し直す。

libpng warning: iCCP: known incorrect sRGB profile

libpng warning: iCCP: cHRM chunk does not match sRGB

まず，cHRM と言うのはPNGの先頭付近にある「チャンク」というメタデータの内の１つで，基礎色度をしめす。この問題の原因はlibpng。[このツール](https://imagemagick.org/index.php)などをつかってメタデータを消すと発生しなくなる。無視しても問題ない。

##### カスケード作成全体を通してのコツ

・ベクトルサイズを小さくても，意外とそれなりの精度がでる。50~80くらいから試すと良い。

・ベクトルサイズを大きくすると精度は多少良くなるが，分類器作成に**膨大な時間**がかかる。

・サンプル画像などのファイル数が多くても処理時間は増える。こちらも少なめから試すと良い。

・サンプル画像作成の際の正解画像は，邪魔な背景などを取り除くのが理想。しかし必須ではない。

・サンプル画像作成のBgColorに注意。誤設定して対象物まで透過したサンプル画像を作ると，精度が著しく下がり，邪魔が映り込んだ元画像の方がむしろ良い。対象物を的確に抽出できる場合においてのみ設定する。

・紛らわしい不正解画像をNEG画像に入れると検出精度は少し上がるが，思いのほか効果は低い。

#### その他特筆事項

##### Web論文より。不正解画像数と誤作動率の対比。

|  |  |  |  |
| --- | --- | --- | --- |
| 不正解  画像数 | 誤作動割合（％） | | |
| Haar-like | HOG | LBP |
| 100 | 100.0 | 11.8 | 100.0 |
| 500 | 38.9 | 4.3 | 40.2 |
| 1000 | 16.1 | 4.3 | 0 |
| 3000 | 10.7 | 6.0 | 0 |
| 5000 | 5.3 | 4.3 | 4.3 |
| 7000 | 15.2 | なし | なし |

この表から分かる事

・不正解画像を増やすことでむしろ誤作動率が上がる事がある。不正解画像も適切に選択する必要がある。

##### GoogleImageから画像の大量ダウンロード

chromedriverをダウンロード

ここからダウンロードできる。Google Chrome のバージョンによってダウンロードするファイルが違ってくるらしい。

<https://chromedriver.chromium.org/downloads>

google\_images\_download を入手

pipでインストールする

pip install google\_images\_download

git で入手する

※GoogleImageの仕様の改変により，本家のソースは使えなくなった。パッチの方を使う事。

ただしパッチ版はパッチ版で --limit が100までしか指定できない。

<https://github.com/hardikvasa/google-images-download>　（本家）

<https://github.com/Joeclinton1/google-images-download/tree/patch-1>（パッチ）

この場合は pythonをつかって動かす事になる。

使い方

-k キーワード --keywords

-l 上限 --limit

-f ファイル形式（拡張子）

-cd ChromeDriver のパスを指定する

（参考資料）<https://qiita.com/Ikko_Kojima/items/4d943c60ff5e886a0544>

### PCで使う（チュートリアル）

#### C#で使う(Win)

NuGetに.NET用にラッパーされたパッケージがあるので，ダウンロードするだけ。

その際複数のパッケージがあるので注意する。

OpenCvSharp3-AnyCPU 　　20.03.24

OpenCvSharp4　が既に出ている20.03.31（これで問題は起こってないのでこれで良い）

.NETで，画像表示ダイアログ（Cv2.ImShow）を使う場合はOpenCvSharp4.Windowsも必要。

.NETでMat(string filename) を使う時は OpenCvSharp4.Windows も必要。

#### pythonで使う(Win)

インストール

python3 -m pip install opencv-python # pipを使う場合

sudo apt install python3-opencv # pip

#### C++で使う（Linux）

##### ・事前準備

sudo apt install cmake build-essential pkg-config # ビルドに必要なもの。

##### インストール

sudo apt install libopencv-dev

##### トラブルシューティング

g++コンパイル時にundefined reference to `cv::String::deallocate()

### ラズパイで使う（チュートリアル）

概要

インストール方法は大きく２つ

#### チュートリアル：ビルド済パッケージ

# 以下は必要パッケージらしい。先にインストールしておく

# C言語(C++) で使う場合

sudo apt install libopencv-dev

# pythonで使う場合

sudo apt install libhdf5-dev libatlas-base-dev libjasper-dev

sudo apt install libqt4-test

python3 -m pip install --upgrade pip # -m pip install -U pip でも良い

python3 -m pip install numpy --upgrade # 最初から入っている事が多いが，upgradeが必要らしい

python3 -m pip install opencv-python==4.1.0.25 # 一部情報ではver4.1.0.25しか動かないとか？

ライブラリは大きく4つ

opencv-python，opencv-contrib-python，opencv-python-headless，opencv-contrib-python-headless

contrib 特殊な用途向けや，実験的なモジュールを含む

headless GUI向きの機能が無いもの

必要なaptパッケージ情報

libhdf5-dev 以下を含む

hdf5-helpers libaec-dev libaec0 libhdf5-103 libhdf5-cpp-103

libhdf5-dev libjpeg-dev libjpeg62-turbo-dev libsz2

libatlas-base-dev 以下を含む

libatlas-base-dev libatlas3-base

（関連）libatlas-doc liblapack-doc 推奨だが，必須ではない

libjasper-dev 以下を含む

libjasper-dev libjasper1

（関連）libjasper-runtime

libqt4-test

#### チュートリアル：コンパイル版

・ソースコードダウンロード

方法は３つ（どれも同じ）

・gitコマンドでリポジトリをDL：

git clone <https://github.com/opencv/opencv>

git clone <https://github.com/opencv/opencv>\_contrib

・githubのzipファイルをDL：

wget -O opencv.zip https://github.com/opencv/opencv/archive/master.zip

wget -O opencv.zip https://github.com/opencv/opencv\_contrib/archive/master.zip

・Webブラウザで公式サイト<https://opencv.org/releases/> へ。  をクリックしてDL（zip形式）

・ビルド

ディレクトリ構成：

~/ ホームディレクトリ

├ opencv/

│ ├ build/ ビルド用のディレクトリ（新規作成）

│ ├ CMakeLists このファイルがある事を確認

│ │ …

├ opencv\_contrib/

terminal：

cd ~/opencv/build # ビルド用のディレクトリへ移動。

# ビルド設定 最後のドット２つ「**..**」を忘れないように。

cmake -D CMAKE\_BUILD\_TYPE=RELEASE \

-D CMAKE\_INSTALL\_PREFIX=/usr/local \

-D INSTALL\_PYTHON\_EXAMPLES=ON \ # python使う予定無くてもONにする

-D OPENCV\_EXTRA\_MODULE\_PATH=~/opencv\_contrilb/modules \

-D ENABLE\_PRECOMPILED\_HEADERS=OFF \

-D BUILD\_EXAMPLES=ON **..**

make -j1 # ビルド。j\*でビルド処理に使うコア数を指定。j4だとエラーになる事もある。※１

sudo make install # インストール

sudo ldconfig # リンカなどの情報の更新？

※１　エラーにこそならなかったが、デッドロック？してるような場面には数回遭遇した。2021-10-22

補足

contribのディレクトリパスの指定方法は ../../opencv\_contribでも良い

オブジェクトファイル(.o) は残る為，ビルド処理は中断しても再開できる。

（公式情報より）<https://docs.opencv.org/master/d7/d9f/tutorial_linux_install.html>

（参考）<https://tomosoft.jp/design/?p=7476>

### C++で使う（共通）

terminal:

find /usr/include/ | grep 'opencv.hpp'

（今のところ有力 2021-10-08）<https://qiita.com/NaotakaSaito/items/f1f1548c8b760629cd26>

（公式）[https://docs.opencv.org/3.0-beta/modules/videoio/doc/reading\_and\_writing\_video.html#videocapture](https://docs.opencv.org/3.0-beta/modules/videoio/doc/reading_and_writing_video.html" \l "videocapture)

（公式のチュートリアル？）<http://opencv.jp/opencv-2svn/cpp/introduction.html>

よくわからんがこれで通った 2021-10-20

g++ $(pkg-config --libs --cflags opencv) main.cpp

Makefileの1例

LIBS\_ffmpeg = -lm -lz -lpthread -lavformat -lavcodec -lavutil

LIBS\_opencv = -lopencv\_core -lopencv\_highgui -lopencv\_imgproc -lopencv\_video -lopencv\_objdetect

LIBS\_autres = -lpthread -ldl -lm

LIBS = $(LIBS\_autres) $(LIBS\_ffmpeg) $(LIBS\_opencv)

CXXFLAGS = -O2 -g -Wall -fmessage-length=0 -D\_\_STDC\_CONSTANT\_MACROS OBJS = test.o FluxVideo.o TARGET = test

$(TARGET): $(OBJS) $(CXX) -o $(TARGET) $(OBJS) $(LIBS)

%.o: %.cpp $(CXX) -o $@ -c $< $(CXXFLAGS) $(LIBS)

all: $(TARGET)

clean: rm -f $(OBJS) $(TARGET)

#### 2021-10-20メモ

sudo apt install pkg-config

pkg-config

環境変数PKG\_CONFIG\_PATHのパスに存在する \*.pc ファイルに記録された情報を元に、ビルドの際に必要な文字列を返す。

export PKG\_CONFIG\_PATH==/usr/local/lib/pkgconfig # 環境変数PKG\_CONFIGの設定が必要らしい

間違っているらしいが

#include <opencv2/opencv.hpp>

#include "opencv2/core/core.hpp"

#include "opencv2/imgproc/types\_c.h"

#include "opencv2/core/version.hpp"

#include "opencv2/highgui/highgui\_c.h"

#include <iostream>

int main(int argc, char\* argv[]) {

IplImage \*Image;

cvNameWindow("DisplayImage",CV\_WINDOW\_AUTOSIZE);

if((Image = cvLoadImage("Image.jpg)) == NULL){

printf("failed to open Image\n");

cvWaitKey (0);

return -1;

}

cvShowImage("DisplayImage",Image);

cvWaitKey(0);

cvDestroyWindow("DisplayImage");

cvReleaseImage(&Image);

return 0;

}

### pythonで使う

インストール

python3 -m pip install opencv-python

ファイル構成

opencv\_test プロジェクトのベースディレクトリ。名前は何でもよい

├ haarcascades

│ ├ haarcascade\_frontalface\_alt.xml GitHubのopencv/opencvからダウンロードしてここに格納

│ などなど

├ main.py メインスクリプト

├ env 仮想環境（任意）

├ requirements.txt 仮想環境のパッケージリスト（任意）

[ main.py ]（サンプル）

import cv2

CASCADE\_FILE\_NAME = 'haarcascades/haarcascade\_frontalface\_alt.xml' # 画像認識に使うカスケードファイル

cascade = cv2.CascadeClassifier(CASCADE\_FILE\_NAME) # 画像認識に使うカスケードオブジェクト

cap = cv2.VideoCapture(0)

try:

while True:

ret, frame = cap.read()

frame\_gray = cv2.cvtColor(frame, cv2.COLOR\_RGB2GRAY) # グレイスケールに変換

detect\_rects = cascade.detectMultiScale(frame\_gray) # 第一引数 image は必須

for rect in detect\_rects:

x, y, w, h = rect # 座標の値を取り出す（任意）

cv2.rectangle(frame, (x,y), (x+w, y+h), (255,0,0), 2)

cv2.imshow('camera capture', frame) # キャプチャした画像を確認する為に停止**【必須】**

if cv2.waitKey(10) & 0xFF == ord('q'): # キーボードで q を押したら終了する

break

except KeyboardInterrupt: # Ctrl+C でも終了できる（任意）

pass

cap.release()

cv2.destroyAllWindows()

[ requirements.txt ]（このサンプルソースを実行した時のpip環境）

numpy==1.21.2

opencv-python==4.1.0.25

pkg\_resources==0.0.0

### Androidで使う

OpenCV for Android

AndroidStudio とjava を使うらしい。

### コマンドレファレンス(Python)

cv2

#### \_\_version\_\_

openCVのバージョンを示す文字列

#### VideoCapture

構文：VideoCapture(device\_num)

構文：VideoCapture(file\_name)

（公式）https://docs.opencv.org/4.0.0/dc/dd3/tutorial\_gausian\_median\_blur\_bilateral\_filter.html

#### VideoWriter

動画の変換などが行える

# 書き出し用のWriteクラスを作成

fmt = cv2.VideoWriter\_fourcc('m', 'p', '4', 'v')

writer = cv2.VideoWriter('./hoge.mp4', fmt, fps, (width, height),0)

while(True):

# 1フレームずつ読み込み

ret, frame = cap.read()

# 動画の最後までいったらretがFalseになるので終了する

if not ret:

break

# 画像をグレースケールに加工

gray\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

# 1フレーム分を書き出し

writer.write(gray\_frame)

# 読み込んだ動画と書き出し先の動画を解放する

cap.release()

writer.release()

#### OpenCV基本

##### destroyAllWindows

flip

反転。二次元配列を垂直、水平、または両軸を指定。

resize

画像を拡大、縮小

##### imread

1コマ分のキャプチャ画像データを読み込む。

構文：

img = cv2.imread("sample.jpg") # 戻り値はnp.ndarray

height, width, channels = img.shape

img = cv2.resize(img, dsize=(width/4, height/4))

##### imwrite

構文：imwrite(filename, mat)

##### imshow

構文：

ウィンドウサイズを変更するには、もとのファイルのサイズを変更すれば良い

img = cv2.resize(frame, dsize=(int(frame\_width\*2), int(frame\_height\*2)))

なお、この後にwaitkeyメソッドを実行しないと、画像キャプチャのウィンドウは表示されない

公式：<https://docs.opencv.org/4.0.0/d4/da8/group__imgcodecs.html>

cmap

<https://docs.opencv.org/4.x/d3/d50/group__imgproc__colormap.html>

##### moveWindow

公式：[https://docs.opencv.org/4.x/d7/dfc/group\_\_highgui.html#ga8d86b207f7211250dbe6e28f76307ffb](https://docs.opencv.org/4.x/d7/dfc/group__highgui.html" \l "ga8d86b207f7211250dbe6e28f76307ffb)

##### release

動画ファイル閉じる、もしくはキャプチャデバイスを終了

##### waitKey

単位はmsec

imshowと併用する。

#### 描写系

##### circle

円を描く

circle(img, (x, y), 5, (0, 0, 255), thickness=-1)

パラメータ

thickness 線の太さ。-1にすると塗りつぶし。

（公式）<https://docs.opencv.org/4.x/de/d09/group__cudaarithm__core.html>

（日本語；注意！v2）<http://opencv.jp/opencv-2svn/cpp/drawing_functions.html>

##### putText

文字を書く

#### 色の処理

##### cvtColor

色空間を交換

公式：[https://docs.opencv.org/4.0.0/d8/d01/group\_\_imgproc\_\_color\_\_conversions.html#ga397ae87e1288a81d2363b61574eb8cab](https://docs.opencv.org/4.0.0/d8/d01/group__imgproc__color__conversions.html" \l "ga397ae87e1288a81d2363b61574eb8cab)

enum（抜粋）

cv2.COLOR\_BGR2GRAY グレイスケール

##### LUT

ガンマ変換を行う。（明るさ変換）

"""

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

【ガンマ補正の公式】

Y = 255(X/255)\*\*(1/γ)

【γの設定方法】

・γ>1の場合：画像が明るくなる

・γ<1の場合：画像が暗くなる

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

"""

# ガンマ変換用の数値準備

gamma = 0.3 # γ値を指定

img2gamma = np.zeros((256,1),dtype=np.uint8) # ガンマ変換初期値

# 公式適用

for i in range(256):

img2gamma[i][0] = 255 \* (float(i)/255) \*\* (1.0 /gamma)

# 読込画像をガンマ変換

gamma\_img = cv2.LUT(img,img2gamma)

情報元：<https://di-acc2.com/programming/python/19009/>

##### equalizeHist

グレイスケール画像のヒストグラムを均一化。

色の変化

（公式）<https://docs.opencv.org/4.0.0/d8/d01/group__imgproc__color__conversions.html>

（公式チュートリアル）https://docs.opencv.org/4.0.0/db/d64/tutorial\_js\_colorspaces.html

threshold 配列の要素に対して一定値で閾値処理を行う。

（公式）https://docs.opencv.org/4.0.0/d7/d1b/group\_\_imgproc\_\_misc.html

（公式チュートリアル）<https://docs.opencv.org/4.0.0/d7/d4d/tutorial_py_thresholding.html>

##### calcHist

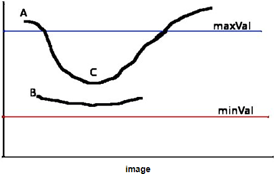
ヒストグラムを表示

#### 画像処理

##### Canny

Canny法によるエッジ検出を行う

Canny法のチュートリアル：<https://docs.opencv.org/4.0.0/da/d22/tutorial_py_canny.html>



cv2.Canny(img\_gray, threshold1=minVal, threshold2=maxVal)

補足

Canny法：エッジ検出に良く使われる

画素値の微分値を計算し、その値がmaxVal(threshold2)以上であれば、正しいエッジと判断。また画素値の微分値がminVal(threshold)以下である場合もエッジでは無いと判断

##### Laplacian

（二次微分フィルタ）によるエッジ処理

公式：<https://docs.opencv.org/4.0.0/d5/db5/tutorial_laplace_operator.html>

[https://docs.opencv.org/4.0.0/d4/d86/group\_\_imgproc\_\_filter.html#gad78703e4c8fe703d479c1860d76429e6](https://docs.opencv.org/4.0.0/d4/d86/group__imgproc__filter.html" \l "gad78703e4c8fe703d479c1860d76429e6)

##### Prewitt

##### Sobel

Sobelフィルタによるエッジ処理

Sobel処理のチュートリアル：<https://docs.opencv.org/4.0.0/d2/d2c/tutorial_sobel_derivatives.html>

##### threshold

ret, threshed\_img = cv2.threshold(img\_gray, 120, 255, cv2.THRESH\_BINARY\_INV)

二値化を行う

公式：<https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html>

[https://docs.opencv.org/4.x/d7/d1b/group\_\_imgproc\_\_misc.html#gae8a4a146d1ca78c626a53577199e9c57](https://docs.opencv.org/4.x/d7/d1b/group__imgproc__misc.html" \l "gae8a4a146d1ca78c626a53577199e9c57)

γ変換

#### アフィン変換

warpPerspectie 透視投影を行う。

補足：アフィン変換

xをax+bに変えるような処理。言い換えると、直線を斜線に変換するような処理を数式で表したもの。

画像の回転、拡大・縮小、平行移動などの処理をイメージした方が理解しやすいかもしれない。

（公式）https://docs.opencv.org/4.0.0/da/d54/group\_\_imgproc\_\_transform.html

（公式チュートリアル）<https://docs.opencv.org/4.0.0/da/d6e/tutorial_py_geometric_transformations.html>

#### 画像の円滑化（ぼかし）

##### GaussianBlur

ガウシアンぼかし。

資料

ガウシアンぼかし

平滑化（ぼかし）処理を行う為の典型的なアルゴリズム

アンシャープマスク処理、エッジ抽出の前処理などに使われる。

##### blur

##### median

画像中に存在する画素値 の中央値で平滑化する。

##### bilateral

エッジを残したまま平滑化する

#### 切り抜き

frame[top : bottom, left : right]

#### フィルタ処理

<https://docs.opencv.org/4.0.0/d4/d86/group__imgproc__filter.html>

画像の円滑化チュートリアル：<https://docs.opencv.org/4.0.0/dc/dd3/tutorial_gausian_median_blur_bilateral_filter.html>

算数的処理？：https://docs.opencv.org/4.0.0/d0/d86/tutorial\_py\_image\_arithmetics.html

#### ノイズ除去

Inpaintチュートリアル：<https://docs.opencv.org/4.x/df/d3d/tutorial_py_inpainting.html>

Eroding and Dilating：<https://docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html>

#### AI関連

##### 物体検出

（good featuresチュートリアル）https://docs.opencv.org/4.0.0/d4/d8c/tutorial\_py\_shi\_tomasi.html

https://docs.opencv.org/4.0.0/d1/de5/classcv\_1\_1CascadeClassifier.html

（matchtemplateチュートリアル）<https://docs.opencv.org/4.x/d4/dc6/tutorial_py_template_matching.html>

##### 特徴点検出

アルゴリズム：SIFT、SURF、ORB、KAZE、AKAZE

<https://docs.opencv.org/3.4/d0/d13/classcv_1_1Feature2D.html>

#### その他

##### set

プロパティ値の設定

cap.get(cv2.CAP\_PROP\_FPS) # 取得

cap.set(cv2.CAP\_PROP\_FPS, 30) # 設定

良く使われるもの？を抜粋

cap.set(cv2.CAP\_PROP\_FOURCC, cv2.VideoWriter\_fourcc('Y','U','Y','V'))

cap.set(cv2.CAP\_PROP\_FRAME\_WIDTH, WIDTH) # 横幅

cap.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, HEIGHT) # 縦幅

cap.set(cv2.CAP\_PROP\_FPS, FPS) # フレームレート

cap.set(cv2.CAP\_PROP\_ISO\_SPEED, 1000)

cap.set(cv2.CAP\_PROP\_HUE # 色相

CAP\_PROP\_AUTO\_WB # ホワイトバランス

CAP\_PROP\_TEMPERATURE # 色温度

公式：[https://docs.opencv.org/4.7.0/d4/d15/group\_\_videoio\_\_flags\_\_base.html#gaeb8dd9c89c10a5c63c139bf7c4f5704d](https://docs.opencv.org/4.7.0/d4/d15/group__videoio__flags__base.html" \l "gaeb8dd9c89c10a5c63c139bf7c4f5704d)

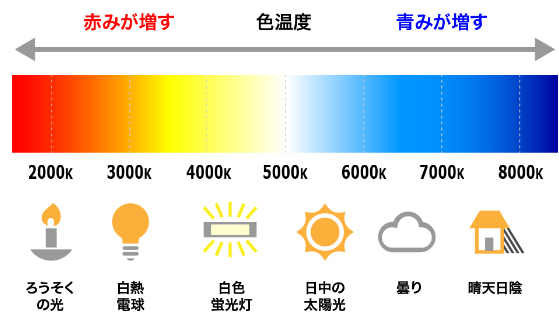
参考：<https://www.klv.co.jp/corner/python-opencv-camera-setting.html>

補足

プロパティはこれで確認できる

props = sorted([(getattr(cv2, p) , p)　for p in vars(cv2) if p[0:8] == "CAP\_PROP"])

色温度について：



### コマンドリファレンス(C#)

OpenCvSharp.Extentionについて

#### Cv2

OpenCVの中核static methodを持つクラス？

##### メソッド

ImShow(string winName, Mat mat) 新規ウィンドウで画像を表示

winName 新規ウィンドウのタイトル（＝Form.Textと同じ）

ImWrite(string filename, Mat mat)

#### Mat

もともとCではMatrixというクラスだったがC++になるときにMatになったらしい。

##### コンストラクタ

Mat () 空のマトリックスを生成する。

Mat (int rows, int cols, MatType type)

rows 画像の縦幅，Heightの事。

cols 画像の横幅，Widthの事。

MatType もともとはint※だが，CVSharp4ラッパーのenumを使う。　※Cでよくあるenum風const

##### メソッド

Empty() Matインスタンス内の配列が要素を持たないかどうか。（Matは複数の静止画も読める）

#### MatType

#### VideoCaptureクラス

カメラ，連続イメージ，動画ファイルなどから動画を取得するクラス。

##### コンストラクタ

VideoCapture ()

VideoCapture (int index, [apiPreference])

int index： 使用するカメラのデバイスIDを指定する。

apiPreference(enum) 使いたいデバイスの背景情報を指定する。（多分指定する事は無い）

bool Read(Mat image) カメラから画像を取得する。

※python では read() -> bool, Mat:

##### プロパティ

CvPtr

FrameHeight

FrameWidth

##### メソッド

bool IsOpened ()

コンストラクタの呼び出しが成功済で，指定したデバイス，またはファイルが使用可能な事を示す。

bool Read(Mat mat)

Grab, decode を実行するらしい。

bool Grab()

ビデオの次のフレームをキャプチャする。失敗するとfalseが返る。

取得したデータはVideoCaptureインスタンス内に取り込まれる為，

Retrieve()

#### CascadeClassifierクラス

カスケード分配器のファイルを用いて画像認識を行うクラス。detectMultiScale() がメインメソッドらしい。

##### コンストラクタ

CascadeClassifier () デフォルトコンストラクタ。多分使う事は無い。

CascadeClassifier (string　fileName)

fileName：カスケード分配器(.xml) のファイル名を指定

##### メソッド

DetectMultiScale(Mat image

double　scaleFactor = 1.1， 縮小処理をする際のステップ。値を大きくすると処理時間が短くなる。

int　minNeighbors = 3， 「検出」とみなす時の近傍矩形の最低数。小さいと誤検出が増える。

HaarDetectionType　flags = 0， 新しいカスケードでは不使用？古いものはcvHaarDetectObjectsと同じ

Size?　minSize = null， 検出対象の最大サイズ。（あまり設定しない気がする）

Size?　maxSize = null， 検出対象の最小サイズ。これより小さいものは無視される

)

scaleFactorとminNeighborsが検出精度を高める上で非常に重要。

#### HaarDetectionType列挙体(enum)

flagsになっていて，検出方法の詳細を指定する。

DoCannyPruning = 1,　(0x0001)

キャニー法によるエッジ検出を行う。エッジが過多，過小な部分の検出を無視する。

顔検出専用の閾値を使い，モデル圧縮，プルーニングによって処理速度改善を行う。

（エッジ検出：画像端を含む四角形範囲に正画像と同じ特徴を検出する事）

ScaleImage = 2, 　(0x0010)

カスケード分配器の検出座標を拡大する代わりに画像の縮小を行う。この値は単一で使用する。

FindBiggestObject = 4, 　(0x0100)

画像の中の最大サイズのオブジェクトを１つだけ検出する。

DoRoughSearch = 8

近似候補が検出されたら，それより小さいサイズの候補を探さない。

FindBiggestObjectを設定，更にmin\_neighbors = 0 である必要がある。

#### NativeMethodsクラス

.NETラッパー専用の静的メソッドの集まりのクラス。元のC++内でのグローバルメソッドへのラッパーと思われる。

##### VideoCapture関連

videoio\_VideoCapture\_operatorRightShift\_Mat((IntPtr obj, IntPtr image)

文字通り　右シフト演算子(>>)のオペレーター。

#### OpenCvSharp.Extensions

BitmapConverter(Mat mat) 文字通りMatをBitmapに変換する。

### オンライン資料

（参考）

公式コマンドリファレンス（英語）： <https://docs.opencv.org/3.4/index.html>

OpenCV 2.2 C++ リファレンス：<http://opencv.jp/opencv-2svn/cpp/>

日本の公式サイト？<http://opencv.jp/>

python http://opencv.jp/opencv-2.2/py/

cpp http://opencv.jp/opencv-2svn/cpp/

その他参考資料

<http://www.ochi-lab.org/course/opencv/flist>

日本語資料：<http://opencv.jp/opencv-2svn/cpp/>

７回にわたって詳しく： <https://www.pro-s.co.jp/blog/system/opencv/6202>

画像加工詳しく：<https://qiita.com/icoxfog417/items/53e61496ad980c41a08e>

公式情報の日本語訳

<http://labs.eecs.tottori-u.ac.jp/sd/Member/oyamada/OpenCV/html/py_tutorials/py_tutorials.html>

### トラブルシューティング

#### Assertion fctx->async\_lock failed at libavcodec/pthread\_frame.c:173

#### Could not load the Qt platform plugin "xcb"

メッセージ全文

`qt.qpa.plugin: Could not load the Qt platform plugin "xcb" in "" even though it was found.`

解決策1

sudo apt install libxkbcommon-x11-0

参考：<https://qiita.com/momomo_rimoto/items/83917d3f9f5dd35457e1>

解決策2

QtとOpenCV双方にGUIがあるのが問題、とか

pip uninstall -y opencv-python

pip install opencv-python-headless

## その他

### Caffe

オープンソースの深層学習用フレームワーク。

BVLC；Berkeley Vision and Learning Centerによる開発

C++で実装。C++/Python/MATLABで使用可能。

公式：http://caffe.berkeleyvision.org/

### Chainer

Preferred Networksが開発。日本初のフレームワーク。

初心者向きで、簡単なネットワークから複雑なdeep learningまでカバー。

利用人口が少ない点や、処理速度がやや遅いのがデメリット。

ver7を最後にPyTorchに移行していく事が発表された。

（公式チュートリアル）<https://tutorials.chainer.org/ja/tutorial.html>

### ReNom

正体不明2022-06-13

<https://www.renom.jp/ja/notebooks/tutorial/basic_algorithm/adam/notebook.html>

### AutoAugment:

概要

ディープラーニングを使ってディープラーニング学習用の画像データを増やす

RandAugment

https://arxiv.org/abs/1805.09501

### HALCON

1か月の評価用ライセンスも存在する。

開発ライセンス 開発時に必要。（ドングル）

ランタイムライセンス 実行時に必要。（ドングル／MACアドレス）

### Keras

ケラス。近年ではtensorflowに内包されている。

ディープニューラルネットワークを使った高速な実装ができる。

## 資料

### 公式チュートリアル

<https://chromium.googlesource.com/external/github.com/tensorflow/tensorflow/+/r0.10/tensorflow/g3doc/tutorials/mnist/beginners/index.md>

Sequentialモデルを用いる方法とFunctionalAPIを用いる方法の違い

https://qiita.com/note-tech/items/bfbd2d63addd490d58ed

### トラブルシューティング

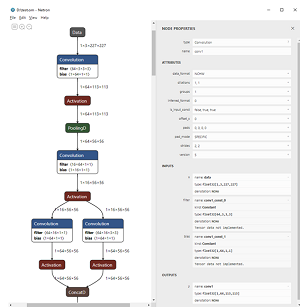
#### qt.qpa.plugin: Could not find the Qt platform plugin "wayland"

export QT\_QPA\_PLATFORM=xcb

# ソフトウェア（補助）

## NETRON

概要

モデルファイルからモデル解析を行う

<https://netron.app/>

## gstreamer

マルチメディア系の処理を簡単にするためのフレームワーク。

公式: http://gstreamer.freedesktop.org/

ライセンス：LGPL

sudo apt install libgstreamer1.0-0 gstreamer1.0-tools

## oneDNN

ディープラーニングのフレームワーク向きのオープンソースライブラリ。

畳み込み、プーリングと言った演算を高速化させる

# ハードウェア

## PC用グラフィックボード

### TITANX

タイタンX

<https://www.nvidia.com/ja-jp/geforce/products/10series/titan-x-pascal/>

### rtx4070

https://www.nvidia.com/ja-jp/geforce/graphics-cards/40-series/rtx-4070-4070ti/

## 外付けGPU

条件として、マザーボードがThunderboltに対応している必要がある



参考：<https://www.asus.com/jp/support/FAQ/1042843/>

Thunderbolt 4 増設ボードも世に存在するが1,2万円はするので、それならGPU カードを買ったほうが良い。

## CUDA

### 概要

CUDA（Compute Unified Device Architecture：クーダ）

NVIDIAが開発・提供しているGPU向きのGPU向けの汎用APIセット。

AMDのCPUには非対応らしい。

公式の対応情報：[https://developer.nvidia.com/cuda-gpus#compute](https://developer.nvidia.com/cuda-gpus" \l "compute)

TensorFlowを使う際はTensorFlowのバージョンによりCUDAのバージョンが異なり、CUDAのバージョンによってbazel（ベイゼル）のバージョンも異なる

TensorFlow公式の対応表：[https://www.tensorflow.org/install/source?hl=ja#gpu](https://www.tensorflow.org/install/source?hl=ja" \l "gpu)

抜粋

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TensorFlow | Python | コンパイラ | bazel | cuDNN | CUDA |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| 2.7.0 | 3.7-3.9 | GCC 7.3.1 | 3.7.2 | 8.1 | 11.2 |

補足：tensorflow1系

バージョンの確認方法

cat /usr/local/cuda/version.txt

### GPUの確認方法

Linux

lspci | grep -i nvidia #

# nvidia-smi を使う場合

nvidia-utils-450

nvidia-smi # 要インストール

windows

ファイル名を指定して実行：dxdiag

### バイナリ

配置場所：/usr/local/cuda/bin/

補足：パスを通す必要があるかも

export PATH=/usr/local/cuda/bin:$PATH

#### nvcc

CUDAのバージョンなどの確認ができる

## OpenVINO

Intelが提供する高性能なコンピューター・ビジョンやディープラーニングを開発可能が可能になるソフトウェア

## シングルボードコンピュータ

TinkerBoard

# 画像関連

## 物体追跡

### 概要

別名：MOT；Multiple Object Tracking

物体検出においては、フレーム間での検出物体の関連付けがなければ、全体の個数や移動方向を判別する事は不可能。

注意点としては、追跡の演算はCPU負荷が重くなりがちなので、演算は軽量だが効果の高いアルゴリズムを採用するようにする。

End to End 検出と追跡の両方を行うもの

2stageモデル 追跡のみを行い、検出は別モデルで実現する

### 主流手法

#### 概要

centroid

検出物体の中心点をとらえ、ユークリッド距離（普通の距離）の一番近いものを同一物体とする。

Kernel-Based

等方性カーネルに基づいた空間マスキング（？）

ヒストグラムに着目した特徴量把握？

IEEE論文：<https://ieeexplore.ieee.org/document/1195991>

Correlation-Based

IEEE論文：<https://ieeexplore.ieee.org/document/9231974>

template maching

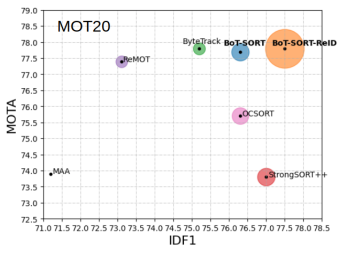
Re-identification

フレーム間での物体の特徴量が一番近いものを同一物体とする

#### 比較

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 発表 |  | 概要／備考 |
| BoT-SORT | 2022年6月 |  | YOLOv8で標準 |
| ByteTrack | 2021年10月 |  | End to End　 YOLOv8で標準 |
| CorrTracker |  |  |  |
| CSTrack |  |  |  |
| FairMOT | 2020年8月？ |  |  |
| TransMOT |  |  |  |
| OCSORT |  |  |  |
| StrongSORT | 2022年2月 |  |  |
| ReMOT |  |  |  |

参考資料



情報元：<https://arxiv.org/pdf/2206.14651.pdf>

SiamMOT

Transformer

#### BoT-SORT

Robust Associations Multi-Pedestrian Tracking

発表：2022年6月

arxiv：<https://arxiv.org/abs/2206.14651>

参考

Github：<https://github.com/NirAharon/BoT-SORT>

#### ByteTrack

従来方式では低いconfidence値のボックスは削除される為にトラッキングが上手く行かない事が多かったが、これらの問題を解決したもの、らしい

・YOLOと併用する事が多い

発表：2021年10月

arxiv：<https://arxiv.org/abs/2110.06864>

参考

Github：<https://github.com/Kazuhito00/yolox-bytetrack-mcmot-sample>

さまざまな手法

#### DeepSort

非常に有名な追跡ソフト？でYOLOと共に使われる事が多い。

#### FairMOT

比較的古い手法？2023年9月現在

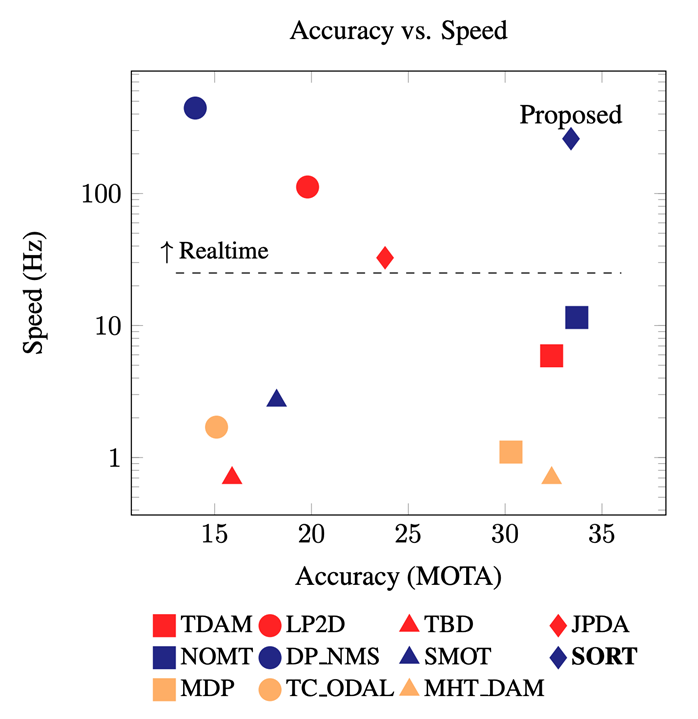
arxiv：<https://arxiv.org/abs/2004.01888>

#### Re-identification

物体追跡の手法のうち、例えば「人」といった同一カテゴリ（クラス）に分類された検出物体を、さらに特徴量ごとに分類する。フレーム間でその特徴量が一番近いものを同一物体とする

<https://qiita.com/nonbiri15/items/2f96e89f57f5fdf37bfd>

#### 資料



情報元：

<https://arxiv.org/abs/1602.00763>

Deep Sort 以降の手法

<https://tech.acesinc.co.jp/entry/2021/11/24/165628>

### 関連するpythonパッケージ

#### motpy.py

非常にシンプルで簡単に使えるMultiple Object Tracking ライブラリ。

pipで利用可能

from motpy import Detection, MultiObjectTracker

tracker = MultiObjectTracker(dt=0.1)

# TOP N とスコアの値によって不要な要素を排除（任意）

scores = scores[:5]

det\_objs = zip(boxes, classes, scores)

for box, class\_id, score in det\_objs:

tracker.step(detections=[Detection(

box=box,

score=score,

class\_id=class\_id,

)])

# アクティブな追跡物体を取得

for obj in tracks = tracker.active\_tracks()

print(obj)

dt step timeを秒単位で指定

詳細設定

tracker = MultiObjectTracker(dt=0.1,

model\_spec={

'order\_pos': 1, 'dim\_pos': 2, # position is a center in 2D space; under constant velocity model

'order\_size': 0, 'dim\_size': 2, # bounding box is 2 dimensional; under constant velocity model

'q\_var\_pos': 1000., # process noise

'r\_var\_pos': 0.1 # measurement noise

},

matching\_fn\_kwargs={

'min\_iou': 015,

'multi\_match\_min\_iou': 015

},

)

#### multitracker.py

OpenCVのcotribにある

<https://github.com/opencv/opencv_contrib/blob/master/modules/tracking/samples/multitracker.py>

### ベンチマーク

#### 概要

ベンチマークと各手法のランキング

<https://motchallenge.net/results/MOT20/>

#### 評価指数

MOTA ；Multi-Object Tracking Accuracy

DF1；

評価指数のまとめ：<https://www.ariseanalytics.com/activities/report/20201016/>

## 画像生成

　深層学習を用いた画像生成手法は概ね以下の3つに大別される。

GAN 生成は早いが、多様な出力1が苦手

Diffusion Model 多様な出力ができるが、生成に時間がかかる

VAE

## 画像処理

|  |  |
| --- | --- |
| 色変換 | カラー画像をグレースケール画像にしたり、BGR→RGBなど |
| 幾何変換 | 画像を拡大・縮小したり、回転させたりする |
| しきい値処理 |  |
| 平滑化 | ぼかしを入れる |
| モルフォロジー変換 |  |
| エッジ検出 |  |

## カメラ関連

### 照明

|  |  |  |
| --- | --- | --- |
|  | 直接光 | 拡散光 |
| 概要 |  | 全体的に明るくする |
| 輪郭 | 強調する |  |
|  | 物体の凸凹、立体的な特徴 |  |
| 注意点 | 強い陰影を生む可能性がある |  |
|  |  |  |
|  |  |  |

光の当て方には以下の３つがある

|  |  |
| --- | --- |
| 光の当て方 | 説明 |
| 正反射 | 入射角と反射角が等しく反射される光がでる当て方  光が拡散しないので強い光がでる |
| 拡散反射 | 入射光がさまざまな方向に反射される光がでる当て方  光が拡散するので弱い光がでる |
| 透過 | 対象物を透過した光がでる当て方 |

スポットライト

ストロボ

### 用語（カメラ関連）

#### ONVIF；Open Network Video Interface Forum

読み方：オンビフ

異なる会社が作った電子デバイスを制御する為のプロトコル、およびその管理団体の名前。

映像データの取得、PTZ制御などが行える。

（アクシス、ボッシュ、ソニーが立ち上げたネットワークカメラ製品のインターフェースの規格標準化フォーラム）

#### GOP；Group Of Picture

MPEGにおいて定められている動画を構成している最小の単位構造の事。

Iピクチャ 単独で復号(デコード)し画像が生成できる静止画そのもの。

Bピクチャ 双方向予測；過去、未来両方のフレームから予測したデータとの差分をエンコードしたもの。

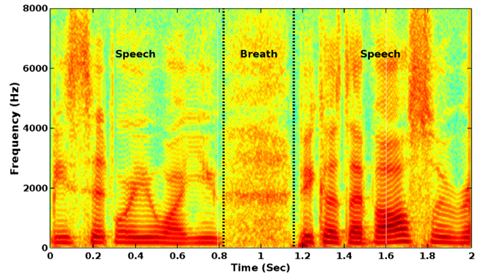
Pピクチャ 一方向の動き補償予測；過去のフレームから予測したデータとの差分をエンコードしたもの

# 音声処理

## 用語

スペクトログラフ；sound spectrograph

別名：声紋、



特定の時間範囲内の周波数特性を可視化したもの

|  |  |  |
| --- | --- | --- |
| 項目 | 意味 | 備考 |
| 横軸 | 時間 |  |
| 縦軸 | 周波数 | 上に行くほど音が高い |
| 色 | 音量（振幅）の大きさ |  |

補足

spectrum は波長

<https://own-search-and-study.xyz/2017/10/27/pythonを使って音声データからスペクトログラムを作/>

# 自然言語処理

## Transformer

<https://deepsquare.jp/2020/07/transformer/>

<https://arxiv.org/abs/1706.03762>

## BERT

## ChatGPT

### APIキー

概要

ChatGPTのアカウント( ChatGPT Plus Subscription )とは無関係

#### APIキーの発行

https://platform.openai.com/overview

右上のPersonal > API keys > +Create new secret key

#### APIキーでできること

##### pythoで利用

pip install openai

最もシンプルな例

import os

import openai

openai.api\_key = os.getenv("OPENAI\_API\_KEY")

response = openai.ChatCompletion.create(

model="gpt-3.5-turbo",

messages=[

{"role": "system", "content": "You are a helpful assistant."},

]

)

print(response['choices'][0]['message']['content'])

公式：https://platform.openai.com/docs/api-reference/authentication?lang=python

https://platform.openai.com/docs/api-reference/completions/create

https://platform.openai.com/docs/api-reference/chat/creates

openai.Model.list()

##### Node.jsで利用

npm install openai@^4.0.0

##### RESTAPIで利用

##### VSCode – Genie AI

### ChatGPT Plus Subscription

・非常に高い言語能力

・写真が画像からもテキストを生成できる

・プラグイン

「WebPilot」

　アクセスできるURLを指定することで、それを踏まえ要約

　PDFファイルやテキストファイル、XMLファイルなどの情報を読み取り要約

「Show Me Diagrams」

### そのほかopenaiのAPI

DALL·E 自然言語プロンプトが与えられた場合に画像を生成および編集できるモデル

Whisper 音声をテキストに変換できるモデル

Embeddings テキストを数値形式に変換できるモデル

Moderation テキストが機密性の高いものであるか安全ではないかを検出できる微調整されたモデル

## 用語（自然言語系）

BoW；Bag of Words

各文書の形態素解析の結果をもとに出現回数をカウントしたもの

TF-IDF；Term Frequency-Inverse Document Frequency

すべての文書に出現する単語と、一部の文書にしか出現しない単語を区別するための方法

TF １つの文書の中に現れる前端後の合計出現回数の内、１つの単語がどれだけの割合で出現したか

IDF ある単語が出現する文書が文書全体のなかでどれだけの割合を占めていたか

生成系

## LangChain

言語モデルを利用してアプリケーションを開発するためのフレームワーク

# 統計学

## 基本

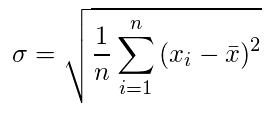
### 標準偏差

standard deviation

母集団データ（サンプル）のばらつきを表わす指標。量記号σ（小シグマ）。

また、σ2の事を「分散」と呼ぶ。

平均±2σ（1.96倍という情報も）の範囲内に入るデータは95%、平均±3σだと99.7%。



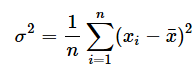
補足：偏差(xn – μ；xの平均)の二乗平均平方根。

（正の値にする為に２乗平方）

python ：numpy.std() でddof=0

エクセル：STDEVP関数

### 分散



偏差平方和 [　Σ(xi – x)2 ]を使いやすくしたものと考える

### 不偏標準偏差

別名：標本標準偏差

sample standard deviation

母集団から一部の標本を抜き取り、母集団のばらつきを見る為の指標。

また、s2の事を「不偏分散」と呼ぶ。

python ：numpy.std() でddof=1 エクセル：

### 標準化

「平均を0、分散を1」にする。統計で良く使われるのはこちら

最大値及び最小値が決まっていない場合や外れ値が存在する場合に利用。

μ 平均 xn – μ ⇒ 偏差

σ　標準偏差：偏差の二乗平均平方根

要するに、偏差を標準偏差で割る。

補足：偏差の絶対値を取っても良いように思うが、二乗する事が多い

python

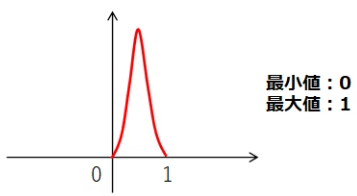
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

data\_scaled = scaler.fit\_transform(data)

### 正規化

別名：最小最大正規化

「最小値を0、最大値を1」にする

最大値及び最小値が決まっている場合などに利用

python

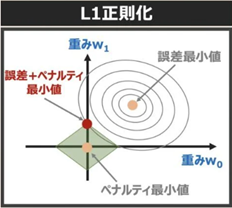
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler([0,1]) # 0~1の正規化

### 正規化

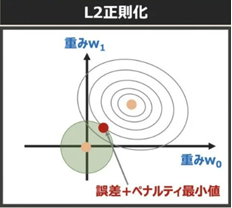
正規化はモデルが過学習を起こさないようにする方法

具体的には特定のパラメータ(重み)にペナルティを科すための追加情報(バイアス)を導入する。

正則化なし（損失関数）

L1正規化

L2正規化



### 偏差値

= z値=（偏差を標準偏差で割る）

z値は特徴のスケールを揃える時に使う（非常によく使われる）

z値はデータが平均の時0になる

70以上 上位2%

60以上 上位20%

50 平均

40以下 下位20%

30 以下 下位2%

例：

from scipy.stats import zscore

standard\_score = 10 \* zscore( df['col1'] )

## 外れ値

先に補足。平均値と言うのは偏りがあった場合にほとんど役に立たない為，統計学では何の意味も持たない。

厳密には「外れ値」と「異常値」は異なるもの：

外れ値 他の値から大きく離れたデータ。

異常値 外れ値のなかでも、大きく離れている原因がわかっているもの。

実用上は区別できないらしい。

### 統計学基本用語

偏差 (deviation)

特定の値から平均値を引いたもの。 [ ]と言った形で表現される。

標本 母集団の中から抽出したn個のデータの事。

### 四分位範囲（IQR；箱ひげ図）

最小値，第1四分位点， 中央値，第3四分位点，最大値 を用いた五数要約。

中央値 データを小さい順に並べた時の中央の値。**平均値とは異なる**ので注意。

四分位点 データを小さい順に並べた時の25%の場所に位置する値。

第1四分位点～第3四分位点の範囲内であれば正常値とみなし，それ以外は異常値とみなす。

⇒ 一般的には最も簡単な方法だろうが，PLCやマイコンなどで実装するのは結構手間になる。

（補足）最近では高校で箱ひげ図を学ぶらしい

### 検定（Smirnov-Grubbsの棄却検定）

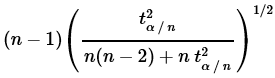
μ±2σ内に入るデータ個数は全体の95% なのでこれで外れ値検定しても良い。（μ±3σ を使う事もあるらしい）

※これを「有意水準5%で検定する」と言う。

また，次の式で求められる値を「検定」と呼び，統計数値表から求めた有意点 の範囲内であるかどうかで評価できる。正規化とも呼ぶ？

τ = ⇒ （評価対象データ － 標本平均）÷ 標準偏差

例えば標本数20で有意水準5%の時の有意点は表によると2.557なので，検定の値が±2.557を超えれば外れ値。



検定の有意点は右の式で算出できるらしい。（※２）（両側と片側で異なる？）

<http://aoki2.si.gunma-u.ac.jp/lecture/Grubbs/Grubbs-table.html>

※２ 近似的な上側 100 α% 有意点は，tα/n を自由度 n−2　の t分布の上側 100 α / n% 点としたとき，とある。

外れ値　続き：（更なる統計学）

## エクセルで出来る統計

### 基本

#### 分析ツールの有効化

エクセルのメニュー：ファイル → オプション →アドイン　分析ツールを選択

→ 「設定」を押す→「分析ツール」にチェック

### 要約統計量（基本統計量）

データ → データ分析 →「基本統計量」を選択

・入力範囲を選択

・出力先を選択

・「統計情報」にチェック

### 資料

EXCEL分析事例

<https://learn.datascience.co.jp/set/1070>

## 管理図

### 概要

|  |  |  |
| --- | --- | --- |
| 分類 | 管理図名 | 特徵 |
| 計量管理図  長さや重量などの連続データ | Xbar-R管理図 | サブグループサイズが8以下の連続データを使用して、 工程の平均と変動 を監視するために使用される。 管理限界線の算出にサブグループ範囲値が 使われる |
| Xbar-S管理図 | 最もよく使われている管理図。 サブグループサイズが9以上の連続データ を使用して、工程の平均と変動を監視するために使用される。 管理限界線 の算出にサブグループ標準偏差が使われる |
| I-MR管理図 | X-Rs管理図とも呼ばれている。 サブグループサイズが1の時、つまりサブ グループというものが存在しない時に使われる |
| 計数管理図  欠陥または不良の時に使用 | P管理図  Laney P 管理図 | 測定結果が 「合格、 不合格」 のような2値の値に対する不良品の比率を監 視するために使用される。 データの分散が大きかったり、 小さかったりす る場合は、 Laney P管理図が使われる |
| NP管理図 | サブグループサイズが一定の時の2値の不良を監視するために使用される。 P管理図と異なり、不良率ではなく不良数を監視する |
| U管理図 | 1つの製品に複数の欠陥が含まれる場合に、 ユニットあたりの欠陥の数を 監視するために使用される。 データの分散が大きかったり、 小さかったり する場合は、 Laney U管理図が使われる |
| C管理図 | サブグループサイズが一定の時の欠陥を監視するために使用される |
| 時間重み付き管理図  工程の小さなずれを検知 | 移動平均管理図 | 移動平均を使って、 工程平均における小さなずれを監視するために使用さ れる |
| EWMA管理図 | 最低値と最高値の影響を取り除いて、 工程変動を監視するために使用され る。工程平均の小さなずれを検知しやすい特徴がある |
| CUSUM管理図 | 目標値とサンプルの偏差を累積和した値が監視に使用される。 累積和で監 視するため、工程平均の小さなずれを検知しやすい特徴がある |
| 多変量管理図  複数変数を同時に監視 | T二乗管理図 | 2変数以上を同時に評価監視するために使用される。 変数間の相関が高 い時に使われる |
| 一般化分散管理図 | 2変数以上の工程分散が正常に管理されているか同時に監視するために使 用される。 変数間の相関が高い時に使われる |
| 多変量EWM  管理図 | 複数の変数に対する、 最低値と最高値の影響を取り除いて、 工程変動を監 多変量EWMA管理図 視するために使用される |
| 希少事象管理図  発生頻度が少ない希少事象を監視 | G管理図 | 発生頻度が低い希少事象を監視するために使用される。 P管理図、 U管理図 と異なり大量のデータを必要としない特徴がある |
| T管理図 | G管理図と同様であるが、 事象間に時刻データがあるか、 経過時間がある 場合に使用する |

参考：<https://www.minitab-kke.com/post/typeofcontrol>

### P管理図

P：不良率 Pn：不良個数 n：サンプル数 k：測定回数

不良率（P）=Pn/n

中心数(CL) は測定回数間の不良率の平均値： = ∑Pn／k

上方管理限界(UCL)：＋　　下方管理限界(LCL)：－

※P+√ の代わりに P+3√ を用いるものもあるらしい。⇒ 違いは何？

1から引くのは，ここで扱う値がパーセンテージだから？

（参考）<https://sigma-eye.com/2018/07/09/what-sigma/>

## 用語

### 標準誤差

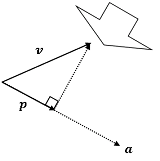
=標準偏差÷ ⇒ 平均値の信頼性を表す。

### 偏差値

平均点が50点・標準偏差が10点になるように調整した時の成績の良さを示す指標。

τ =10 × +50

### 射影

平面上の図形のすべての点と、その平面外の一点とを結ぶ直線をひくこと

参考：射影ベクトル（右図）

### 正規分布（ガウス分布）

標準偏差の分布図が中央値を中心に，左右対称になる分布の事。（右図参考）

標本のデータ数が1000を超えると正規分布に近くなる可能性が高い。

正規分布の場合，中央値±σ（標準偏差）に入るデータ個数は全体の約68%，±2σ内に入るデータ個数は全体の95%になる。

なお、この式は f(x) =

μ：平均　　σ：標準偏差

### スミルノフ･グラブス検定

## 資料

プログラマーのための統計学

<https://qiita.com/Esfahan/items/bd9e52837bbaeb168b9f>

統計手法について

<https://www.statweb.jp/method>

EXCEL分析事例

<https://learn.datascience.co.jp/set/1070>

# 数学

## 平均

### 相加平均

arithmetic mean 別名：算術平均

日常生活で用いる、一般的な平均の事

### 相乗平均

geometric mean 別名：幾何（きか）平均

GDPの成長率などに用いられる。記号Π（大パイ）

例：

### 対数平均

log-average

調和平均

harmonic mean、subcontrary mean

H ＝

## 確率

### 組み合わせ

C

### 順序

P

### 大数（だいすう）の法則

law of large numbers

試行回数を極限まで増やすと、平均値は期待値に収束する法則の事。

### 平均二乗誤差

線形回帰などの際、「誤差」を取り除く為に誤差を算出する手法の内の一つ。

誤差には負の値も想定される為に2乗してすべて正の値に変換する。

θ0：切片 θ1：傾き θ：傾き yk ：目的変数

補足：数学用語では誤差はerrorと呼ばれるらしい。「ノイズ」と呼ぶ人もいるらしい。

### 確率密度関数

ある連続変数が特定の範囲内の値となる確率。積分を用いて、面積的に算出する。

正規分布の確率密度関数は

μ：平均 σ2：分散

### 確率分布

probability distribution

### 確率変数

離散確率変数 サイコロの目のように、変数が飛び飛びで存在しているもの ⇒ 確率

連続確率変数 1.1111、1.010101といった様々な変数をとる⇒ 確率密度

### 尤度関数（ゆうどかんすう）

Maximum likelihood function

サンプリングしてデータが観測された後、そのデータは元々どういうパラメーターを持つ確率分布から生まれたものだったか、という問いに答える為のもの。

### 共分散

covariance

2つの変数の関係の強さを表す指標の一つ。２つ変数の偏差の積を平均したもの。

共分散が正であるときは、一方の値が増加するともう一方の値が増加する傾向にあり、これを正の相関と呼ぶ。共分散が負である時は、一方の値が増加するともう一方の値が減少する傾向にあるといえ、これを負の相関と呼ぶ。

### 相関係数

2 つ以上の複数の変数の間の共分散を示す。標準偏差の比較なので、単位が異なる2つ以上の変数の比較ができる。-1以上+1以下の実数値で、**-1または+1に近いほど関係性が強い**。0であれば無関係。

説明力：相関係数を二乗した時の値

例：相関係数が0.32だった　説明力=0.322 =0.1004 10%くらい

補足：Sxy：xとyの共分散 Sx：xの標準偏差 Sy：yの標準偏差

### 有意差

仮説と標本の観察による結果の差が「誤差の範囲内」なのか「誤差では済まされない意味のある差」なのかの指標となる値。

### 標本分散行列

2つの変数の（標本）分散（σ2）の行列。

例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 点数 | A | B | C | D | E | 平均 |
| 国語 | 50 | 50 | 80 | 70 | 90 | 68 |
| 数学 | 50 | 70 | 60 | 90 | 100 | 74 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 偏差 | A | B | C | D | E | 平均 |
| 国語国語 | 324 | 324 | 144 | 4 | 484 | 256 |
| 国語数学 | 432 | 72 | -168 | 32 | 572 | 188 |
| 数学数学 | 576 | 16 | 196 | 256 | 676 | 344 |

この場合の標本分散共分散行列 →

python：numpyのcov, ddof=0 ddof=1：不偏分散共分散行列

### 共分散行列

別名：相関行列

2つの変数の相関係数で作成した行列。

python：numpyのcovに標準化されたlistを引数, ddof=0

pandas

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.corr.html

NumPy

https://numpy.org/doc/stable/reference/generated/numpy.corrcoef.html

seaborn

https://seaborn.pydata.org/examples/many\_pairwise\_correlations.html

scikit-learn

<https://scikit-learn.org/stable/modules/generated/sklearn.covariance.empirical_covariance.html>

### 切断正規分布

せつだんせいきぶんぷ

確率変数 x の定義域が有限な確率分布。上下とも有界 (A ≤ x ≤ B) なものを二重に切断された正規分布、どちらか一方だけのものを単一切断正規分布という。

## 対数

### 基本

値nを何乗するとmになるか、を算出する。

例：

この例では2を何乗すると8になるか、の算出で、答えは3。

この時の2を**底**と呼ぶ。

底

自然対数 底がネイピア数の対数

常用対数 底が10の対数

### ネイピア数

別名：自然対数の底、オイラー数（紛らわしいのであまり使われない）

e=のeはEuler（オイラー）からきているとか。

使い方

math.e

math.exp(1)

### 指数

expx = ex

exは何度微分しても、あるいは何度積分しても同じ形のままなので都合が良い事がある。

## 行列、ベクトル

### ベクトル

#### 概要

いくつかの数をまとめて表現したもの。例えば身長と体重の場合となる。

横方向に「行ベクトル（横ベクトル）」と呼ぶ

縦方向に「列ベクトル（縦ベクトル）」と呼ぶ

https://atarimae.biz/archives/23930　行列のかけ算

<https://atarimae.biz/archives/24166>　ベクトル

メモ

数字をN個並べればN次元のベクトルになる

例：3次元ベクトル

その意味では、ベクトルは1列複数行の行列とほぼ同じ。

固有方程式 |A−λI|=0

＝ ＝ ＝

＝ (4−λ)(3−λ) - (1)(2) → (4−λ)(3−λ)＝2→ λ=

<https://www.headboost.jp/matrix-eigenvalues/>

#### 距離

|  |  |  |
| --- | --- | --- |
|  | マンハッタン距離 | ユークリッド距離 |
| 概要 | 軸に沿った距離 | 直線距離 |
| ノルム | L1ノルム | L2ノルム |
|  |  |  |
|  |  |  |

L1正規化、L2正規化と関係がある？

#### ノルム

ベクトルの大きさをスカラーで表現したもの

距離とかなり近い概念。（最初は＝距離と考えると良い）

具体例：a=(x1,y1)の時

### 疎行列

別名：sparse matrix, sparse array, スパース行列

成分のほとんどが零である行列のこと。

CSR形式

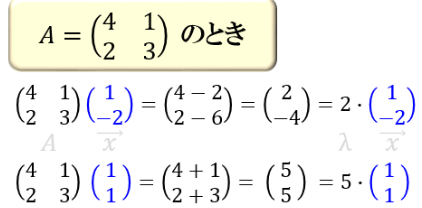
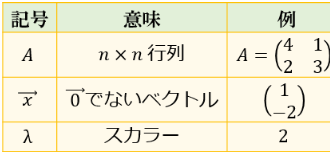
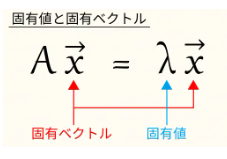
例えばnumpyの[ [1, 0],[0, 1] ] を (0,0) 1, (1,1) 1 とデータが1なっている場所の行と列インデックスで表現するもの。

C00表現？

### 固有ベクトル

### 記号

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 記号 | 意味 | 呼び方 | 例 | 備考 |
| A | n×n行列  （行数=列数) | 正方行列 | A= |  |
|  | でないベクトル | Aの固有ベクトル | 列ベクトル | xTとも記述される？ |
| λ | スカラー | Aの固有値 | 2 | 対角行列とも呼ばれる？ |



### 固有値分解

正方行列を固有値と固有ベクトルに分解する手法

・行列の対角化によって行われる

・行列の特性を調べることができる

### 用語

定数倍

以下のような事を定数倍と呼ぶ（らしい）

の時

ドット積

アダマール積

Hadamard Product

同じサイズの行列に対して成分ごとに積を取ることによって定まる行列の積

成分

1つの要素の事

内積

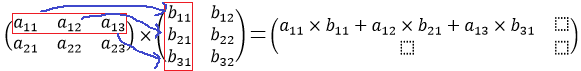
２つの行列同士の積の和

行列Aの列数と行列B の行数が等しくなければならない

出来上がる行列は [行列Aの行数、行列Bの列数] となる

左上から順に計算。

行列Aの1行目と行列Bの1列目の内積が、解の1行1列目の成分。



具体例

参考：<https://w3e.kanazawa-it.ac.jp/math/category/gyouretu/senkeidaisu/henkan-tex.cgi?target=/math/category/gyouretu/senkeidaisu/gyouretu-no-seki.html>

転置行列

→

転置行列に関する法則

補足：tは通常右上(At)に付けられるが、乗数などと区別して左上に付ける書籍もある。

## 分布

### 二項分布

binomial distribution

離散型の分布。「勝率55%のチームが7選した時の勝ち数の確率」といったものを算出する。

離散型：10回コインを振って表が出た回数が3.5回、といった値を３回、４回とする事

ポアソン分布

Poisson distribution

発生する可能性が低い事象が起こる回数ごとの確率を求める。

例えば一年間の市内の交通事故死者の数が１人になる確率、２人になる確率などをそれぞれ求める。

カイ二乗分布

例えばさいころを50回振って1が30回出たとすると確率的に何かおかしい感じがするが、それを分布として表現するもの。

## 関数、変数

### 基本

説明変数

y=ax+b（など）の時、xの事

目的変数

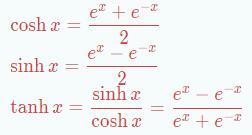
y=ax+b（など）の時、yの事

条件

例：Xが0を超える場合は0、0以上1以下の場合は1、0未満の場合は0

### 三角関数

#### ハイパボリック

別名：双曲線関数

ハイパボリックコサインなどと呼ばれる。

シャインエックスなどと呼ぶ人もいるとか。

指数関数をもとに定義される三角関数の事。

## 記号

### ギリシャ文字系

μ アルファベットのMにあたる。平均(mean) を意味する。

Σ 総和を示す

μ アルファベットのMにあたる。平均(mean) を意味する。

Σ 総和を示す i×jの総和、iが1〜9、jが1〜9まで。

### 上付き文字など

ハット、キャレット、circumflex。分野により意味は異なる。

・点推定量を差す。

・何かしらの拡張。

### 演算子

要素、集合。 例 xは集合Xの要素である。

例： 要素xが集合Sに属している

向きが違うだけ？

積集合（共通部分）　例：A={1,2} B={2,3} の時、

和集合（全部まとめたもの）例：A={1,2} B={2,3} の時、

自然数の集合

整数の集合

有理数の集合

実数の集合

複素数の集合

### その他

実数、とくに複素数における実数部を表す。

目的変数（response variable）

ニアリーイコール；近似。

偏微分。特定の文字以外を定数とみなして微分したもの。

$ (順に)階乗, 超階乗

ファイ（小）：オイラー関数 φ(n)として使用される。

ファイ（大）：Φxでxn-1における既約円分多項式

円分多項式：1のべき根に関連のある多項式

│ 用途により意味が異なる。

{x|x<2} 定義

P(A|B) 集合： Bという条件のもとでAとなる確率

例P(t=1|x) xを与えた時t=1になる確率

（参考）<https://ja.wikipedia.org/wiki/数学記号の表>

（参考）<https://zenn.dev/wsuzume/articles/b0b3a51cac5d7fe4555b>

## その他

### ベルヌーイ分布

「成功か失敗か」「表か裏か」「勝ちか負けか」のように２種類のみの結果しか得られないような実験、試行（ベルヌーイ試行）の結果を0と1で表した分布を指します。

### 共線性

Collinerarity

入力変数xに対して出力変数yが線形に強く相関している状態の事

## 用語（数学）

行列

3次元以上の変数データを一括で処理するのに便利なツール。

この時の123や456を「行」と呼び、14や25などを「列」と呼ぶ

行列を構成する数字1つ1つを「成分」、行が m ・列が n の行列を「m 行 n 列の行列」と呼ぶ。

＋= みたいな事ができて便利。

切片

せっぺん；英: intercept

座標平面上の曲線などと座標軸の交点の事。x軸との交点を x切片、y軸との交点を y切片と呼ぶ。

要素

：xが0または1の時は　の意味

arg arg max f(x) ={1}　といった形で記述。意味はf(x)の最大値を達成するxは1。

オイラー関数 正整数 n に対し1から n までの整数のうち互いに素なものの個数を与える関数

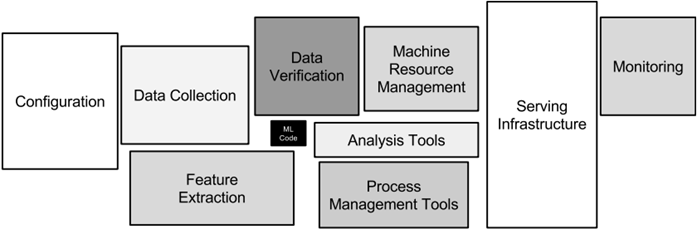
円分多項式

# 資料

## AIプロジェクトにおいて重要な事

資料

AIシステムにおいてはAIモデル作成は；ｌほんの一部であり、他の部分が重要であることを発表した論文より：



<https://proceedings.neurips.cc/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf>

## パターンマッチ

<https://www.visco-tech.com/technical/guidance/patternmatch/>

## IPCアルゴリズム

<http://www.sanko-shoko.net/note.php?id=3c5m>

参考：https://data-analysis-stats.jp/深層学習/深層学習の最適化アルゴリズムまとめ

## AI白書

毎年角川が出版している、AIの現状について網羅的にまとめた本

## 関連する試験

### DeepLearning for ENGINEER；E資格

JDLA；Japan Deep Learning Association

ディープラーニングの理論を理解し、適切な手法を選択して実装する能力や知識を有しているかを認定する

・理論やアルゴリズムなどに関する知識を問う

・そもそも認定プログラムの受講が受験資格として必須になる

・実装スキルはそれほど無くても合格できるとか

合格率は74%

それでも**AI関連の資格の中では最難関**らしい。

ある程度知識のある人しか受けないから、という分析もあるが…？

公式：<https://www.jdla.org/certificate/engineer/programs/>

### Python 3 エンジニア認定データ分析試験

数学の基礎知識、pythonライブラリを利用し、統計学的課題を用いた解決能力を問う試験。

・AIというより、データ分析に焦点

・pythonライブラリについて問う内容が大半（全体の6,7割）

正答率70％以上が合格 合格率は86% 学習時間は30時間程度

割と簡単らしいが？

公式：[https://cbt.odyssey-com.co.jp/pythonic-exam.html#python3cda](https://cbt.odyssey-com.co.jp/pythonic-exam.html" \l "python3cda)

### TensorFlowデベロッパー認定資格

TensorFlowを用いてディープラーニングに関する問題を解決する能力を評価するビギナー向きの試験。

（TensorFlow:Googleが開発したディープラーニングのフレームワーク）

難易度：G検定以上、E資格未満

・公式に推奨されているCourseraのオンライン講座が存在する

・試験はPyCharm上で実施するのでなれておく必要がある

・試験の実施はすべて英語

### Deep Learning For GENERAL ；G検定

JDLA；Japan Deep Learning Association

ジェネラリスト検定

ディープラーニングを事業に生かす知識を有しているかどうかを確認するため試験

⇒ G検定はビジネス向け，どちらかというとプロジェクトマネージャーとか向き？

合格率は66.17％ 割と簡単らしい

公式：<https://www.jdla.org/certificate/general/>

### DS検定リテラシーレベル

データサイエンティスト検定 リテラシーレベル

SQL、統計学、データサイエンスのビジネスへのアプローチ方法など

⇒ あくまでビッグデータの取り扱い？

ER図などデータベースを扱うのに必要な事柄も含まれるらしい

### Jetson AI Specialist

NVIDIAのベンダー資格

オンライントレーニングコースを受講し、Jetson Nanoを用いたプロジェクトをレポートして提出して認められたら合格。⇒ なんとなく、微妙

コース

・Jetson AI Fundamentals　時間：4時間程度　費用：？

<https://developer.nvidia.com/ja-jp/embedded/learn/jetson-ai-certification-programs>

補足　こんなコースもある

・Jetson NanoでAIを始める(Getting Started with AI on Jetson Nano)　時間：8時間程度　費用：無料

<https://courses.nvidia.com/courses/course-v1:DLI+C-RX-02+V1-JA/>

### 統計検定2級

5等級ある「統計検定」の資格の中でも実践的な資格

完全に統計学に焦点を当てた試験。

大学基礎レベルの統計学の知識と問題解決力が問われます。

### 数学検定1級

AIで用いる統計計算の更に基礎部分である，固有値，固有ベクトルなどの知識を問う

# 用語

## 重要用語

### バウンディングボックス

画像編集ソフトや文書作成ソフトなどで、図形や画像、文字列などのオブジェクトの周囲を囲む矩形の境界線

（関連）セグメンテーション

### IoU；Intersaection-Over-Union

ひとつ前のフレームの同一物体と共有するバウンディングボックス領域の重なり。

### 過学習

別名：過剰適合；Overfitting

学習（トレーニング）データに過剰に適合すること。学習中の精度は高く見えるが、未知のデータに対する精度が極めて低い状態。学習での性能がとても良いのに、テストでの性能が悪い。

## AI関連用語

### 機械学習

人工知能による分析、アルゴリズム作成全体を包含する言葉。

機械学習の１つとして深層学習がある、といった位置づけ。

狭義には、決定木のような原始的な学習アルゴリズム群を差す。

### 深層学習

Deep Learning

膨大なデータを基に分析を行い、精度の高い予測・判断を実現する。

３層以上のニューラルネットワークは線形にclass分けを行う？

### Affine

全結合層。入力を共通にした複数の人工ニューロンの入力から活性化関数の入力までの事。活性化関数は含まない。

= () + () w：重み付け b：バイアス

### Sparse Modeling

少ない情報から全体像を復元しようとする科学的モデリング手法。

### top-k accuracy

物体認識の評価基準で、モデルが予測した分類のうち、確率の高いもの上位k位までに正解がある確率

よく使われるのがtop-1, top5。

### ハイパーパラメータ

モデル学習の設定項目のこと。

（学習時に更新されるのは重みのみ）

### XAI；Explainable AI

分類器を逆コンパイルする事で、特定のAIの分類手法を可視化したりする事。

AIが下した結論の根拠を示すことが主な目的。分類器の性能向上の為の分析にも使える。

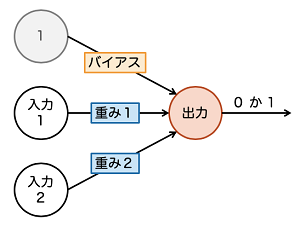
### 入力変数

特徴量(feature)、予測変数(predictor)、独立変数(independent variable)とも呼ばれる。

### 説明変数

### パーセプトロン

現在の深層学習の基礎部分となる、1958年に発表された人間の脳神経回路を真似た学習モデル。

バイアス（b） ：１を出力する度合の調整値。通常、負の値？

重み（w） ：入力値の重要度の調整値。例：入力１は0.7、入力2は0.3

ノード ：右図の〇の部分。ニューロンとも呼ぶ。

手順

|  |  |
| --- | --- |
| 1. 特徴ベクトルを用意 | 入力データ（説明変数）をベクトル化したもの |
| 2. 係数ベクトルを用意 | 決定境界線の傾き。重み。 |
| 3. 識別関数を用意 | 特徴ベクトルと係数ベクトルの内積 |
| 4. 活性化関数 | 結果を出力する |

### ミニバッチ学習

全ての画像を分けられるように重みを徐々に変えていく「バッチ学習」と言うのがあるが、全ての画像に対する勾配を同時に扱う為、GPUのメモリ消費が大きく現実的でない。

その為、5万枚中100枚など、画像をランダムで抜き取って学習をするミニバッチ学習を行うのが一般的。

### ワンホット表現

特定の要素の値のみが1で、その他全ての要素の値が0となっている配列。

### 学習率

値が小さいほどモデルのパラメータ更新が遅く、値が大きいと収束せず最適解が導き出せない。

学習の最初は大きく、最後の方は小さくなる事が理想。学習率スケジューラというものを用いると、そのような学習率の制御が行える。

他にも、バッチサイズ、エポック数、重みの初期値などが学習率に影響を与える。

SGD=0.01, Adam=0.001

### アンサンブル

ensemble

弱識別器を複数あつめて一つの識別器を作成する手法。

アンサンブル学習、アンサンブルメソッドという言葉がある。

主に以下の手法がある

バギング

ブースティング

（関連）ランダムフォレスト

隠れ層

ニューラルネットワークにおいて、入力層でも出力層でも無い層の事。

## その他用語

### ONNX

異なるフレームワーク間でモデルを共通化して使う仕組みの１つ。

# Memo

ベクトルの大きさ（距離）

right\_bottom = sorted(targets, key=lambda arg: \

np.linalg.norm(arg.box.reshape(2,2)[1] - base.reshape(2,2)[1]))

⇒ 失敗

失敗理由：

ベースのright bottom に一番近いのが右下ピンとは限らないらしい

中点座標法

バウンディングボックスの中点の座標x値、y値

# X, Y の中点を求めるラムダ式

middle\_pt = lambda pt\_min, pt\_max: pt\_min + ((pt\_max - pt\_min)/2)

# 対象のリストを左上から順に並べる

left\_top = sorted(targets, key=lambda arg: \

(middle\_pt(arg.box[2],arg.box[0]), middle\_pt(arg.box[3],arg.box[1])))

# 対象のリストを右下から順に並べる（マイナス１を掛ける）

right\_bottom = sorted(targets, key=lambda arg: \

(middle\_pt(arg.box[2],arg.box[0])\*-1, middle\_pt(arg.box[3],arg.box[1]) \* -1))

⇒成功！！

このあと、IDの重複がないことを確認する（重複だけ見ればよい）

補足１：

簡易的には(x\_min, y\_min) でも良いかもしれないが、バウンディングボックスの大きさによってはうまくいかない可能性が考えられる。中点での比較が無難。

Numpy

.ndim Number of Dimension？階数＝配列次元数を返す